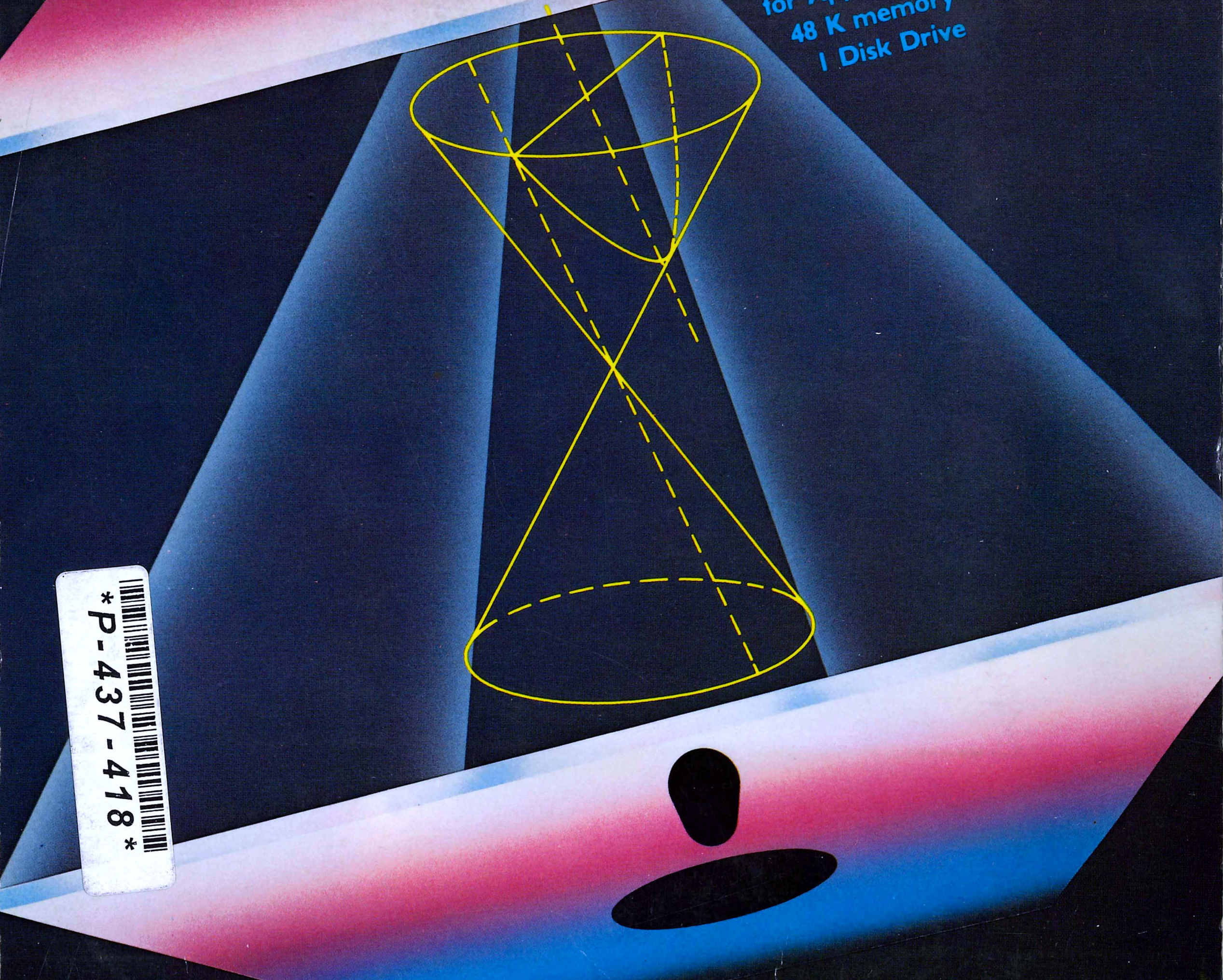
Jon L. Sicks

INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS

INCLUDES TEXT AND SOFTWARE
for Apple II series computers
48 K memory
1 Disk Drive



P-437-418

NOTICE TO CONSUMERS

THIS BOOK MAY NOT BE RETURNED FOR CREDIT OR REFUND
IF THIS SEAL HAS BEEN BROKEN OR TAMPERED WITH.

P-H Software

DISK ENCLOSED

INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS

Mrs. Waymire,
I'll always remember
your dedicated, inspiring
Teaching.
Jon

APPLE COMPUTER, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE.

DOS 3.3 Developer's System Master and DOS TOOLKIT are copyrighted programs of Apple Computer, Inc. licensed to Jon L. Sicks to distribute for use only in combination with INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS. Apple Software shall not be copied onto another diskette (except for archive purposes) or into memory unless as part of the execution of INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS. When INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS has completed execution, Apple Software shall not be used by any other program.

If your diskette is defective or damaged in transit, return it directly to Prentice-Hall at the address below for a no-charge replacement within 90 days of the date of purchase. Mail the defective diskette together with your name and address to:

Prentice-Hall, Inc.
Attention: Ryan Colby
College Operations
Englewood Cliffs, New Jersey 07632

The author and publisher of this book have used their best efforts in preparing this book and software. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS

JON L. SICKS

Department of Mathematics/Statistics
University of Massachusetts
Amherst, MA

PRENTICE-HALL, INC., Englewood Cliffs, NJ. 07632

Library of Congress Cataloging in Publication Data

Sicks, Jon L. (date)
Investigating secondary mathematics with computers.

Includes index.
1. Mathematics—Data processing. 2. Mathematics—
Computer programs. I. Title.
QA76.95.S53 1985 510'.28'54 85-3534
ISBN 0-13-503152-4

Editorial/production supervision: **Karen Skrable Fortgang and Lisa Schulz**
Interior design: **Anne Bonano**
Cover design: **Photo Plus Art, Celine A. Brandes**
Manufacturing buyer: **John Hall**

© 1985 by **Prentice-Hall, Inc.,** Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-503152-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Syndey*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

To my mother and the memory of my father.

CONTENTS

Preface *xi*

Notes to the Teacher *xv*

1 Stepping *1*

2 Whole Number Division *3*

3 Euclidean Algorithm *8*

4 Divisors *12*

5 Primes *15*

6 Perfect Numbers Again *24*

7 Reducing Fractions *26*

8 Fraction Arithmetic *29*

9 Means *34*

10 Harmonic Mean of Fractions *43*

11 Square Roots *49*

12 Farey Sequences and Roots *59*

13 Decimals and Long Division *67*

14 Periodic Decimals *74*

15 Brute-Force Pi *92*

16 Archimedes' Pi *96*

17 Linear Equations I *105*

18 Linear Equations II *113*

19 Linear Equations III *120*

20 Linear Equations and Round-Off Error *127*

21 Lines *134*

22 Graphs *142*

23 Velocity *149*

24 Acceleration *156*

25 Kepler and Newton *162*

26 Parametric Equations *171*

27 Polar Coordinates *178*

28 Limits *197*

29 Derivatives *206*

30 Integrals *215*

31 Root Searches *230*

32 Series *247*

33 Hints and Answers *264*

Index *275*

CATALOG OF CLASS DISK

HELLO
STEPPING
MULTIPLES
WHOLE NUMBER DIVISION
GCD DEMO
DIVISORS
SIEVE DEMO
SIEVE
PERFECT SEARCH
REDUCE
FRAC ARITH
MEANS
FRACTION SUBROUTINES
SQUARE ROOT DEMO
SQUARE ROOTS REPEATEDLY
ORDER
SORTER
FRACTION SORTER
FRACTION ROOT SEARCH
MEDIANT ROOT SEARCH
FRACTION BRACKETER
LONG DIVISION
LD AUTO BREAK
EQUATION GETTER
EQUATIONS
EQUATIONS CLEAR
XY EQUATIONS
LINES
GRAPHER
VELOCITY
ACCELERATION
KEP 1
HERON'S DEMO
PARAMETRIC EQUATIONS
POLAR POINT PLOTTER
POLAR GRAPHER
POLAR CONICS
LIMIT TESTER
DERIVATIVE LIMIT
DERIVATIVE FUNCTION
INTEGRAL LIMIT
INTEGRAL FUNCTION
BISECTION
NEWTON'S
BISECTION GENERAL
GEOMETRIC SERIES
SINE SERIES
SINE SHIFTER

PREFACE

The microcomputer, with its lightning-fast computations and excellent graphics, is a powerful tool for teaching and learning mathematics. However, the traditional mathematics curriculum was not designed with the computer in mind. Therefore, today's mathematics teacher must have new ways to look at traditional topics in order to use the computer for their illumination. Of even more importance is the need to present nontraditional areas of mathematics that are suitable for the secondary school and that were inaccessible before the advent of the computer.

This book and the accompanying Class Disk provide a computer-connected treatment of many topics from the secondary mathametics curriculum currently being used in most secondary schools as well as several nontraditional areas of mathematics. These range from elementary topics, such as the greatest common divisor, to those involving complicated programming, such as solving systems of linear equations, and those involving advanced mathematics, such as the integral and infinite series. Several chapters involve computer graphics. In these, the student learns, for example, to use the computer to graph a function and to model motion.

The focus throughout is on mathematics, not programming. However, there is an illuminating interplay between the two. In some areas, such as number theory, using the computer leads to discoveries in mathematics. In others, such as round-off error and series, mathematics unlocks some of the mysteries of the computer.

This package is designed to be used as (1) the text in a mathematics course for prospective and in-service secondary teachers, (2) the text in a high school computer-mathematics course, or (3) a resource for secondary mathematics teachers who want to use microcomputers in their classes.

These materials are designed for use by students working singly or in teams on AP-

PLE microcomputers. The programs seldom require 48K memory and will run on any APPLE II+, IIe, or IIc.

Each chapter generally begins with a simple program from the Class Disk for the mathematical task at hand. After we have analyzed the listing to see how the program works, the student runs the program to see its results. Then the student upgrades the program to make it faster, more accurate, or to perform additional tasks. To improve the program in this fashion, the student must understand the program and the mathematics involved.

Occasionally we discuss a mathematical technique and ask the student to write the entire program to perform it. In these cases, we make explicit the operations the program must perform, and, if the program is to be more than a few lines long, we provide a flow chart.

Although programming is required of the student, a beginning knowledge of BASIC is sufficient. To start off, the student should understand FOR-NEXT loops and IF-THEN statements. Some familiarity with arrays and string variables will help, although the chapters in which these terms first appear can be used to introduce them.

There is a section of notes for the instructor which contains a discussion of each chapter's contents and a description of the optional Solutions Disk. At the end of the text is a section of hints and answers for many of the exercises in the frontmatter.

I take this opportunity to thank those who helped in this project: James Hedlund, Loren Henry, and William Snyder for their suggestions; the mathematics faculty at Amherst Regional Junior and Senior High Schools for testing a preliminary version; the participants in the University of Massachusetts' Summer Programs on Computers and Mathematics for their working through the materials; finally, those at the University of Massachusetts who funded these summer programs: the President of the University, the Dean of the Faculty of Natural Sciences and Mathematics, and the Director of the Division of Continuing Education.

HOW TO BOOT THE DISK

In order to do the exercises in this book, you must use the programs on the Class Disk. The disk boots up in the standard way. With the computer turned off:

1. hold the disk with the lable side up, your thumb on the label;
2. insert the disk in the disk drive (drive number one if you have two);
3. close the drive door;
4. turn on the monitor; and then
5. turn on the computer.

The disk drive's in-use light will come on, the drive will whir, and the title of the disk will be displayed on the screen. Then the catalog of the disk will be shown on the screen. Hit RETURN a couple of times to get to the end of the catalog. Now you are ready to LOAD or RUN the programs that the exercises require.

HOW TO SAVE YOUR PROGRAMS

Many of the exercises ask you to modify a program from the Class Disk. If you have worked on a program for a while and want to save your program to work on later, do the following: Choose some name for your program different from the names of the other programs on the disk (the original Class Disk programs are locked so you can't accidentally erase them). Suppose you name your program MY PROG. Type SAVE MY PROG. The computer will put a copy of your program on the disk under the name MY PROG. Then, when you want to return to that program, you can LOAD or RUN it.

If you save many long programs, you may run out of room on the label side of the disk. If the disk doesn't have enough room for your program, the message DISK FULL will appear in response to the SAVE command. If this happens, you can still have your program on the other side of the disk: remove the disk from the disk drive, turn the disk over, and insert it with the label side down. Now SAVE your program in the usual way.

WHEN YOU ARE FINISHED

When you are finished working with the Class Disk, turn off the computer and remove the disk from the drive. If you do not turn off the computer before trying to use a different disk, the computer may refuse to load or run programs. This is because of the copy protection built into the Class Disk.

NOTES FOR THE TEACHER

You can obtain the Solutions Disk, which contains solution programs for most of the programming exercises, from Prentice-Hall. With these programs available, you may provide the solution program to a student who gets stuck on the programming, so that he or she may proceed to subsequent exercises. In addition, you might occasionally want to skip a programming problem altogether, give the students the solution program, and go on to the exercises that use the program.

In the following chapter notes, we discuss the content of each chapter, its prerequisites, and the subsequent chapters that depend on it. We also point out where the various elements of BASIC first appear.

CHAPTER 1, STEPPING

This chapter studies the effect of round-off error caused by repeated additions. It seves as a warning to the unwary programmer and should be completed before any of the other chapters. The program uses FOR-NEXT, IF-THEN, INPUT, and HOME.

CHAPTER 2, WHOLE NUMBER DIVISION

This chapter introduces the ideas of quotient and remainder in whole number division. It is preparatory to Chapter 13, ''Decimals and Long Division.''

The program uses the commands VTAB, GET, and a POKE command to protect the instructions from scrolling. We discuss the function INT and the student uses it in Exercise 2-10.

CHAPTER 3, EUCLIDEAN ALGORITHM

This chapter introduces the Euclidean algorithm, which will be used in many programs in which we must reduce fractions.

Exercise 3-3 introduces a research-discovery question. Have a contest to see who can find the pair giving the largest number of steps. Exercises 3-4 to 3-17 give step-by-step guidance toward a program to solve the question posed in Exercise 3-3. Although you could leave this chapter after Exercise 3-3, the surprise that awaits in Exercise 3-17 is worth the effort: The pairs of numbers are neighboring Fibonacci numbers! (This is part of Lame's theorem in number theory.)

CHAPTER 4, DIVISORS

This chapter examines perfect, deficient, and abundant numbers. The student modifies a divisors program to list these kinds of numbers. This chapter is a prerequisite to Chapter 6, ''Perfect Numbers Again.''

CHAPTER 5, PRIMES

This chapter introduces the sieve method with a program that shows the sieving on the screen. The programs in this chapter use one-dimensional arrays to hold the tables of numbers. The program SIEVE DEMO uses subroutines, but the student doesn't have to confront them directly.

After improving the original sieve program in Exercises 5-2 and 5-3, the student writes a program to compute prime decompositions in Exercise 5-5.

Exercise 5-8 compares finding greatest common divisors by prime factorization to using the Euclidean algorithm. In Exercise 5-9, the student writes a direct (nonsieve) program to compute a table of primes. Exercise 5-10 gives an opportunity to improve the sieve program. It is fun to have a race here between sieving and direct method programs. The Solutions Disk contains my entry, FAST SIEVE.

The chapter ends with some research-discovery questions (Exercises 5-11 to 5-14) involving primes.

CHAPTER 6, PERFECT NUMBERS AGAIN

Here we return to the search, begun in Chapter 4, for perfect numbers. We ask the student to use the prime factorization program developed in Chapter 5. You could instead provide the student with the program PRIME FACTORS from the Solutions Disk.

CHAPTER 7, REDUCING FRACTIONS

This chapter begins the work with fractions, which will continue in several later chapters. The reducing program uses the Euclidean algorithm of Chapter 2. It also uses a subroutine to prepare the student for the later extensive use of subroutines.

CHAPTER 8, FRACTION ARITHMETIC

This chapter completes the development of a program to perform fraction arithmetic which was begun in Chapter 7. We give the student a program consisting of subroutines to compute the greatest common divisor, reduce a fraction, multiply two fractions, and read a fraction from the keyboard. We also use these fraction arithmetic subroutines in several later programs. After analyzing the given subroutines, the student writes subroutines to add fractions and print their sum. Finally, the student writes a driver program to complete the fraction arithmetic program.

The fraction reading subroutine works with string variables and the functions LEN, VAL, MID$, LEFT$, and RIGHT$.

CHAPTER 9, MEANS

This chapter introduces the arithmetic, geometric, and harmonic means. This topic is continued in Chapter 12, ''Farey Sequences and Roots,'' and in Chapter 16, ''Archimedes' Pi.''

In Exercise 9-4, the student writes a program to compute these means for two numbers. Then Exercises 9-6 to 9-8 present applied problems that the student can solve with the program of Exercise 9-4.

After this work with the means of two numbers, we give the definitions of the means of several numbers and a program to compute them. Then come three applied problems, each solved by one of the means. Finally, the student is presented with discovery questions (Exercises 9-11 to 9-13) involving the three means.

CHAPTER 10, HARMONIC MEAN OF FRACTIONS

Here we continue the work with the harmonic mean begun in Chapter 9. The decimal computations of Chapter 9 obscure the properties of the harmonic mean, so the student writes a program to compute the harmonic mean as a fraction. We provide the subroutines to do the fraction arithmetic and outline how to use them in computing the harmonic mean. With the resulting program, the student continues with the harmonic mean discovery question from Chapter 9.

CHAPTER 11, SQUARE ROOTS

This chapter starts with a demonstration that the computer's SQR function does not really give the square root of a number. This reviews the meaning of square root and raises the question of how we might approximate a square root. We address this here by approximating square roots by fractions. We do this in a very naive, inefficient manner: Make an array of all the fractions with denominator up to some bound, sort them with a bubble sort, and search for the two whose squares lie closest on either side of the number whose square root we want. This slow method prepares the way for more efficient ones to come in Chapter 12, "Farey Sequences and Roots," and Chapter 31, "Root Searches." The program that sorts fractions is also used to construct Farey sequences, which appear in Chapter 12.

CHAPTER 12, FAREY SEQUENCES AND ROOTS

Here we develop a simple, efficient method of approximating square roots by fractions. The student sees mathematics used to develop an approximation method using only fractions. This method, by providing an error estimate, is perhaps better than the computer's SQR function.

We suggest in the text that the student use the fraction arithmetic program from Exercise 8-7. You could instead provide the program FRACTION ARITHMETIC from the Solutions Disk.

Exercises 12-13 to 12-16 are applied square root problems in which the approximation is required, somewhat naturally, to be a fraction.

CHAPTER 13, DECIMALS AND LONG DIVISION

This chapter explains what is meant by an infinite decimal expansion for a fraction. It also shows that the long division algorithm is a natural and efficient way to compute the digits in the expansion. Chapter 2, "Whole Number Division," is a prerequisite. The discussion of expansions is continued in Chapter 14, "Periodic Decimals."

Exercises 13-1 to 13-3 use a simple program to bracket fractions. We use bracketings to determine, from the definition, the first few digits in decimal expansions. Exercises 13-4 and 13-5 use a mathematical idea to make more efficient use of the program in finding decimal digits. Exercises 13-6 to 13-11 incorporate this idea into the program itself. Exercises 13-12 and 13-13 use the improved program and show that it can give only a few digits of the decimal expansion because the variables get too large for the computer to handle. Exercises 13-14 to 13-21 gradually modify the program in order to keep the variables small. In the end, the program gives correct digits forever, and we show that it is doing exactly the long division algorithm.

CHAPTER 14, PERIODIC DECIMALS

This chapter continues the work with decimals begun in Chapter 13. We use the long division algorithm to investigate terminating decimals, purely periodic decimals, the period of a repeating decimal, and the fact that some fractions have more than one expansion. Finally we investigate expansions with bases other than 10. Expanding $1/10$ in base 2 makes obvious the reason for the round-off error observed in Chapter 1.

CHAPTER 15, BRUTE-FORCE PI

This chapter asks the student to program an inefficient, straight-from-the-definition method of approximating π. This prepares for the more efficient method in the next chapter.

CHAPTER 16, ARCHIMEDES' PI

Here we develop Archimedes' method for approximating π. Although the discussion is somewhat long, it involves only basic geometry and trigonometry. The student programs the ancient algorithm, which is very simple and efficient. The geometric and harmonic means (Chapter 9) appear in a reformulation of the algorithm.

The next four chapters deal with systems of linear equations. We develop programs to manipulate the equations with all coefficients and constants handled as fractions. This eliminates round-off error problems and allows the student to focus on the method for solving a system. Although the entire treatment is in terms of equations, you might want to point out that the operations work only on the coefficients and constants, so that we're really working on the matrix of the system.

CHAPTER 17, LINEAR EQUATIONS I

Here we start with a discussion of the kinds of operations on equations that you could expect a computer to be able to do—adding two equations and multiplying one by a constant. Then we discuss the notation to be used. Two-dimensional arrays must be used to hold the numerators and denominators of the coefficients.

Exercises 17-1 to 17-4 analyze the subroutine which enters the equations. In Exercises 17-5 to 17-7, the student writes a subroutine to print the equations. In exercises 17-8 and 17-9, the student writes a subroutine to muliply an equation by a constant and a subroutine to add one equation to another using the given fraction arithmetic subroutines. In Exercise 17-10, the student completes the program by adding a menu subroutine to give the user a choice of which operation to use on the equations. Finally, Exercise 17-11 presents some systems for the student to solve.

CHAPTER 18, LINEAR EQUATIONS II

In this chapter the student investigates the number of solutions a system may have by solving several systems. The experience gained here in manipulating equation lays the foundation for Chapter 19.

CHAPTER 19, LINEAR EQUATIONS III

Here we lead the student to develop a program to solve a system of equations automatically. We provide a program that can clear a specified variable using a given equation and that can interchange two equations. After using this program to analyze the solving process, the student extends the program to an automatic one.

CHAPTER 20, LINEAR EQUATIONS AND ROUND-OFF ERROR

In this chapter we abandon fractions as coefficients to see the disasters that the round-off error can create.

The next seven chapters use the APPLE's high-resolution graphics.

CHAPTER 21, LINES

Here we introduce graphics in order to draw straight lines. First we investigate the commands HGR, HCOLOR, and TEXT and the function HPLOT. Then we develop a plotter subroutine, which is used throughout the graphics chapters, to convert our usual x-y coordinates system into a u-v system for the computer's screen. This provides an application of equations of straight lines to a computer problem.

After we analyze a program that draws the graph of a line given the slope and y-intercept, the student uses the program to investigate some geometry of lines. This leads to the problem of *scaling*.

Exercise 21-9 asks the student to find the correct scale factor for the monitor and change the program so that lines on the screen have the correct apparent slope. The student must use this scale factor in subsequent graphics chapters.

With the scaled program, the student then solves line-graphing problems in Exercises 21-10 and 21-11, which require a knowledge of straight-line equations.

Finally, the student writes programs to draw lines given the slope and a point on the line, or given two points on the line.

CHAPTER 22, GRAPHS

This chapter starts with a program that uses the plotter subroutine of Chapter 21 to draw the graph of a function. Then the student expands the program to include the drawing of horizontal and vertical lines. The student then uses the expanded program to investigate various functions and to study local maximum and minimum points.

The next four chapters use the graphics screen to model motion.

CHAPTER 23, VELOCITY

This chapter deals with straight-line motion and prepares the student for Chapter 24. Slopes of lines and the Pythagorean theorem come up in the discussion. Vectors do not appear here, but they are about to burst through in the discussion of slope and speed. The student modifes the original program to make the moving point bounce off the edges of the screen.

CHAPTER 24, ACCELERATION

This chapter continues the discussion of Chaper 23. First we add constant downward acceleration to mimic gravity. The student can observe that the resulting paths appear to be parabolas. The student modifies the original program so that the motion is started by specifying the direction and speed rather than the x and y velocities. The last three exercises ask the student to experiment with more exotic acceleration situations.

CHAPTER 25, KEPLER AND NEWTON

This chapter models the motion of a planet subject to the force of gravity. It shows that Newton's law of gravity implies Kepler's three laws of planetary motion. The derivation of the formulas for the program involves vectors. The first attempt at modeling the motion does not produce elliptical orbits, so the student sees that different approximation methods give different results.

To counter possible adulation of computers, Exercise 25-6 poses a simple problem which has no computer solution.

To test Kepler's second law, we need Heron's formula for the area of a triangle. We discuss it and then illustrate it in exercise 25-7.

CHAPTER 26, PARAMETRIC EQUATIONS

This chapter studies motion specified by parametric equations. The student discovers the effect of changing the parametrization. Exercise 26-12 presents the beautiful Lissajous curves. In Exercises 26-13 and 26-14, the student uses trigonometric identities to verify the equation of familiar-looking curves.

CHAPTER 27, POLAR COORDINATES

Here we introduce polar coordinates. We use the graphics screen to plot points and draw graphs specified by polar coordinates. The student discovers the fundamental facts about polar coordinates in Exercises 27-1 to 27-6. Then we turn to drawing the graphs of polar coordinate functions. Here the student can experiment with polar graphs to an extent that is simply impossible with hand computations, pencil, and paper.

Next we present the polar coordinate description of conic sections. Here the student experiments with the eccentricity and directrix to discover facts about these curves and is challenged to find polar functions whose graphs mimic ours.

The next four chapters cover topics involving calculus.

CHAPTER 28, LIMITS

This chapter, which is preparatory to Chapters 29 and 30, ''Derivatives'' and ''Integrals,'' respectively, presents the intuitive idea of limit. We start with a program to list the values of a function near a base point. The student uses the program to investigate several functions and proceeds naturally through one-sided limits, infinite limits, and limits at infinity. We also point out the limitations of the computer in investigating limits.

CHAPTER 29, DERIVATIVES

We start here with a program similar to that used in Chapter 28 to approximate the limit of the difference quotient for a function. With this program the student investigates the definition of the derivative limit. Then, using the difference quotient to approximate the derivative, the student investigates the derivative function. Being able to approximate the derivative at several points enables the student to discover (or verify) standard derivative formulas.

CHAPTER 30, INTEGRALS

Here we define the integral and then use a program that approximates the integral by rectangles. In testing the program, the student discovers that it is slow to converge and converts it to the trapezoidal rule. The student then investigates the integral as a function of the upper limit and discovers properties of the integral and the fundamental theorem of calculus.

CHAPTER 31, ROOT SEARCHES

This chapter completes the discussion of roots begun in Chapters 11 and 12. We start with the bisection method. The student finds that the bisection method is sometimes better than the computer's SQR function. Next we turn to Newton's method, which the student discovers is much faster than bisection at approximating square roots.

After this study of square root approximations, we turn to roots of general functions. The student investigates Kth roots, roots of polynomials, and roots of trigonometric functions. This naturally leads to the arctangent function. The chapter ends with two applied problems whose solutions demand finding roots of trigonometric functions.

Exercise 31-23 calls for the student's graphing program from Exercise 22-4. You could instead supply the program GRAPHER 2 from the Solutions Disk.

CHAPTER 32, SERIES

The goal of this chapter is to approximate the sine and exponential functions by power series to see if that might be what the computer is up to in its SIN and EXP programs. We start with a program to compute the sums for a geometric series. This leads naturally to power series representations for functions.

We use derivative formulas to see how to compute a power series for the sine function. The student finds that, for X near 0, the power series converges to the values for SIN(X), but that it doesn't do so well for X away from 0. Then the student uses identities to make the program shift far away X values near to 0 before computing the power series.

In the same spirit, we develop power series for the cosine, exponential, and logarithm functions. The student discovers that the power series is not very suitable for the logarithm function and that an integral approximation is better. Finally, the student derives and tests a power series for the arctangent.

1

STEPPING

Although the computer is a powerful tool for studying mathematics, it has certain limitations. For example, it cannot add correctly! What may be worse, it sometimes tries to cover up its mistakes by lying about its answers. The program STEPPING (see list below) illustrates this behavior and reveals a bit about how the computer operates.

LISTING 1-1

```
100  REM   STEPPING
110  HOME
120  PRINT "    THIS PROGRAM STARTS AT X = 1 AND"
130  PRINT : PRINT "ADDS 1/N TO X FOR N TIMES, THEN CHECKS"
140  PRINT : PRINT "WHETHER X = 2."
150  PRINT : PRINT "WHAT WOULD YOU LIKE THE DENOMINATOR"
160  PRINT : PRINT "N TO BE";: INPUT N
170 STP = 1 / N
180 X = 1
190  PRINT : PRINT "X = ";X
200  FOR K = 1 TO N
210 X = X + STP
220  PRINT "X = ";X
230  NEXT K
240  PRINT
250  IF X <  > 2 THEN  PRINT "X MISSED 2 "
260  IF X = 2 THEN  PRINT "X HIT 2 "
270  PRINT : PRINT "FOR STEP SIZE 1/";N;" = ";STP
```

Lines 120 to 160 of STEPPING announce what the program does and obtain the step 1/N which will be added to X.

Lines 180 and 190 initialize X to 1 and print this initial value.

A FOR-NEXT loop in lines 200 to 230 add 1/N to X, N times.

Then, in lines 250 and 260, the ending value of X is tested to see whether it hit 2.

Exercise 1-1. Run STEPPING for denominators N = 2, 3, 5, and 10 to see what it does.

Note that for some step sizes, such as ⅓, X hits 2 as we would expect. However, for others, such as ⅒, the computer prints "2" as the value for X, but then admits that X in fact missed 2. So we see that the computer sometimes prints a different value from the one it carries internally. This is because it internally carries more decimal places than it is willing to display on the screen. This discrepancy is something you should be aware of. It can cause difficulty when you write programs, particularly if you try to compare a decimal-valued variable X to some fixed value in an IF-THEN statement.

We have thus seen the computer lying about its misses. But why can it not hit 2 every time? Round-off error causes the missing. The machine can carry only a finite number of decimal places, so it must sometimes round off the step size. Then it adds its (incorrect) value of the step size over and over again to X, so that X eventually has an incorrect value.

The more times an incorrect step size is added to X, the further X should deviate from the expected values. Hence, smaller step sizes, such as 1/23, should result in misses. Larger ones, such as ⅙, might result in hits. The next exercise examines this behavior.

Exercise 1-2. Find and record those fractions 1/N that result in hits, for N up to 40.

Exercise 1-3. Use your results in Exercise 1-2 to help you find two values for N larger than 40 that result in hits.

Why do denominators that are powers of two result in hits? It is because the computer works with numbers in binary (base 2) form. A fraction such as 1/16 has a finite binary expansion, so no rounding-off occurs when the machine uses it for a step size. Hence, the value for X stays true throughout.

There is a way to avoid this round-off error when you want to move a variable along in equal steps. Rather than adding the step value STP to the old X value as STEPPING does, add integral multiples K*STP to the original X.

Exercise 1-4. Modify STEPPING so that it moves X along as described above. Change line 210 to

```
210 X = 1+K*STP
```

Exercise 1-5. Run this modified version of STEPPING for several N. Is it an improvement over the original version?

2

WHOLE NUMBER DIVISION

Let A and B be whole numbers. To divide A by B means to find the largest multiple of B that does not exceed A. For example, to divide 25 by 7, look at the multiples of 7:

$$
\begin{aligned}
7*0 &= 0\\
7*1 &= 7\\
7*2 &= 14\\
7*3 &= 21\\
7*4 &= 28
\end{aligned}
$$

So the largest multiple of 7 that does not exceed 25 is 21. Since 21 = 7*3, we say that 7 goes into 25 three times, or that the *quotient* is 3 when 25 is divided by 7.

The program MULTIPLES (see Listing 2-1) lists the multiples of a user-supplied number. It can be used to find the quotients in whole number division.

LISTING 2-1

```
                    MULTIPLES

100  REM    MULTIPLES
110  HOME
120  VTAB 10
130  PRINT "      THIS PROGRAM LISTS THE MULTIPLES"
140  PRINT : PRINT "OF A USER-SUPPLIED NUMBER."
150  VTAB 23
160  PRINT "      PRESS ANY KEY TO CONTINUE."
170  GET ANS$
180  HOME
```

```
190  PRINT "      FOR WHAT NUMBER DO YOU "
200  PRINT : PRINT "WANT THE MULTIPLES";
210  INPUT B
220  HOME
230  PRINT "TO STOP, TYPE S"
240  PRINT : PRINT "TO CONTINUE, HIT RETURN"
250  POKE 34,5: REM  PROTECT INSTRUCTIONS FROM SCROLLING
260  VTAB 6
270 Q = 0
280  PRINT B;"*";Q;" = ";B * Q
290  GET ANS$
300  IF ANS$ = "S" THEN 330: REM  END
310 Q = Q + 1
320  GOTO 280
330  POKE 34,0: REM  END SCROLLING PROTECTION
340  END
```

Exercise 2-1. Run MULTIPLES to see how it works.

Exercise 2-2. In each of 1 to 5 below, find the largest multiple of B that does not exceed the target number A. What is the quotient when A is divided by B? (Use the program MULTIPLES.)

1. A = 356, B = 17 (Answer: quotient = 20)
2. A = 755, B = 19
3. A = 792, B = 8
4. A = 17632, B = 14
5. A = 314567, B = 237

Modify the program MULTIPLES so that it finds the largest multiple for you (see Exercise 2-3).

Exercise 2-3. Delete lines

```
230 PRINT "TO STOP, TYPE S"
240 PRINT "TO CONTINUE, HIT RETURN"
290 GET ANS$
300 IF ANS$ = "S" THEN 330: REM END
```

Exercise 2-4. Insert lines

```
212 PRINT: PRINT "WHAT IS YOUR TARGET NUMBER";
214 INPUT A
230 PRINT "TARGET = "; A
290 IF A < B* (Q+1) THEN 330: REM END
```

Exercise 2-5. Do Exercise 2-2 again, with this version of MULTIPLES.
Now to return to our example, dividing A = 25 by B = 7. We found that 21 = 7*3

was the largest multiple of 7 not exceeding 25, so that 3 was the quotient. This largest multiple, 21, misses 25 by 4, so we say that the *remainder* is 4. That is, when 25 is divided by 7, the quotient is $Q = 3$ and the remainder is $R = 4$.

This may be summarized as follows:

$$3*7 \leq 25 < 4*7$$

so $Q = 3$.

$$25 - 3*7 = 4$$

so $R = 4$.

In general, to divide A by B, we must find Q, such that

$$Q*B \leq A < (Q + 1)*B$$

Then Q is the quotient, and $R = A - Q*B$ is the remainder.

The program WHOLE NUMBER DIVISION (see Listing 2-2) finds the quotient and remainder for user-supplied whole numbers. Use this program in the following exercises.

LISTING 2-2

```
100  REM    WHOLE NUMBER DIVISION
110  HOME
120  VTAB 10
130  PRINT "      THIS PROGRAM DOES WHOLE NUMBER"
140  VTAB 12: HTAB 17
150  PRINT "DIVISION."
160  VTAB 23
170  PRINT "      PRESS ANY KEY TO CONTINUE."
180  GET ANS$
190  HOME
200  PRINT "      WE WILL DIVIDE A BY B"
210  PRINT : PRINT "A = ";: INPUT A
220  PRINT : PRINT "B = ";: INPUT B
230  HOME
240  PRINT : PRINT "DIVIDING ";A
250  PRINT : PRINT "BY ";B
260  POKE 34,5: REM    PROTECT FROM SCROLLING
270  VTAB 6
280 Q = 0
290  PRINT Q;"*";B;" = ";Q * B
300  IF A < B * (Q + 1) THEN 330: REM     END
310 Q = Q + 1
320  GOTO 290
330  PRINT : PRINT "WHEN ";A;" IS DIVIDED BY ";B
340  PRINT "THE QUOTIENT IS ";Q
350  PRINT "AND THE REMAINDER IS ";A - Q * B
360  POKE 34,0: REM   END SCROLLING PROTECTION
```

Exercise 2-6. In each item below, find the quotient and remainder when A is divided by B.

1. A = 2,175, B = 17
2. A = 3,110, B = 5
3. A = 4,172, B = 3
4. A = 11,793, B = 37
5. A = 21,994, B = 781
6. A = 489, B = 7,016

Exercise 2-7. The teacher has 23 pupils and a bag of 217 apples. The teacher distributes as many apples as possible to the students, making sure that each student gets the same number of apples. How many apples does each student get? How many does the teacher have left over at the end?

Exercise 2-8. There are 1,753 schools in the state. The board of education has 6,532 sample math tests to share among the schools. How many does each school get? How many are left over at the board of education?

Exercise 2-9. The intergalactic police force has 375,246 police stations and 9,563,257 laser blasters to distribute evenly to the stations. How many does each station get? How many are left over at the headquarters?

The program WHOLE NUMBER DIVISION finds the quotient by comparing ever higher multiples of B to A. This can take a long time if A is considerably larger than B. The function INT in BASIC provides a shortcut method of computing the quotient and remainder in whole number division. However, this method uses decimals, and thus is not strictly a whole number operation. Because it is so much faster, we will overlook this excursion out of the realm of whole numbers and use the faster method in subsequent programs.

Here is how to use INT in whole number division: Consider our original example, dividing 25 by 7. As a decimal,

$$25/7 = 3.571428\ldots$$

INT(X) is the largest integer which does not exceed X, so

$$\mathrm{INT}(25/7) = 3$$

which is the quotient. The remainder is

$$25 - 3*7 = 25 - \mathrm{INT}\ (25/7)*7$$

In general, then, in dividing A by B, the quotient is Q = INT(A/B), and the remainder is

$$
\begin{aligned}
R &= A - Q*B \\
&= A - \mathrm{INT}\ (A/B)*B
\end{aligned}
$$

Exercise 2-10. Write a program to find the quotient and remainder for user-supplied numbers A and B using the method outlined above. Do Exercise 2-6 with your program to see how much faster this method is.

3

EUCLIDEAN ALGORITHM

The Euclidean algorithm is an ancient method for finding the greatest common divisor of two whole numbers. We investigate the algorithm here and will use it later in programs that work with fractions.

The algorithm uses repeated divisions. For example, to find the greatest common divisor (abbreviated GCD) of 42 and 15, the method proceeds as follows:

1. Divide 42 by 15:

$$42 = 2*15 + 12$$

The quotient is 2, and the remainder is 12.

2. Divide 15 by 12:

$$15 = 1*12 + 3$$

The quotient is 1, and the remainder is 3.

3. Divide 12 by 3:

$$12 = 4*3 + 0$$

The quotient is 4, and the remainder is 0.

The process continues until a zero remainder is reached. Then the current divisor is the GCD. Thus, the last division above shows that 3 is the GCD of 42 and 15.

In order to write a program to perform this algorithm, we must see how each division is set up from the ingredients of the previous one. Note that in the division steps above, the dividend at one stage is the divisor from the previous stage. Also, the divisor at one stage is the remainder from the previous stage. In terms of variables, this may be expressed as follows: Given dividend A and divisor B, divide A by B, getting quotient Q and remainder R. If R = 0, the GCD is the divisor B. If R <> 0, make the new dividend B (LET A = B), and make the new divisor R (LET B = R).

The following program, GCD DEMO, uses the Euclidean algorithm with variables A, B, Q, and R as described above to compute the greatest common divisor of two user-supplied numbers N1 and N2.

LISTING 3-1

```
100  REM  GCD DEMO
110  HOME
120  PRINT "FOR WHAT NUMBERS DO YOU WANT THE GCD?"
130  PRINT : PRINT "FIRST NUMBER = ";: INPUT N1
140  PRINT : PRINT "SECOND NUMBER = ";: INPUT N2
150 A = N1:B = N2: REM   SET UP FOR ALGORITHM
160 Q =  INT (A / B): REM  DIVIDE A BY B
170 R = A - Q * B: REM  REMAINDER
180  PRINT : PRINT A;"=";Q;"*";B;"+";R
190  IF R = 0 THEN 250: REM   ALGORITHM FINISHED, GCD IS B
200  REM  IF R <> 0 MUST DO ANOTHER DIVISION
210  REM  FIRST, PAUSE:
220  GET ANS$
230  REM  NOW SET UP FOR NEXT DIVISION
240 A = B:B = R: GOTO 160
250  PRINT : PRINT "GCD(";N1;",";N2;")= ";B
```

Note the use of the function INT in line 160 to find the quotient and the computation of the remainder in line 170. This is the shortcut method we discussed in the chapter on whole number division.

Line 180 prints the result of the division.

Line 190 branches on the value of R.

If R = 0, the GCD is printed in line 250 and the program ends.

If R <> 0, the program pauses (line 220). Then, in line 240, it sets up for another division.

Exercise 3-1. Run GCD DEMO for various pairs of numbers.

1. What happens if the first number is zero? If the second is zero?
2. What difference does it make if the numbers are entered in decreasing order rather than increasing?

Exercise 3-2. Eliminate the pause in line 220, and run this version for various pairs of numbers.

Exercise 3-3. How many steps does the algorithm require to get to the GCD? Find the pair of numbers between 1 and 999 that requires the largest number of steps.

To try to solve Exercise 3-3 by trial and error is hopeless. A systematic search is needed, and we'll use the computer to do the searching. The next few exercises develop the program. Although it takes several modifications to the current program, the programming experience is valuable and a mathematical surprise awaits at the end.

Exercise 3-4. Use a variable COUNTER to count the number of steps required to get the GCD. Insert line

```
175 COUNTER = COUNTER+1
```

When the GCD has been found, print the number of steps taken. Insert line

```
260 PRINT: PRINT "NUMBER OF STEPS = "; COUNTER
```

Try this version to be sure it counts correctly.

Exercise 3-5. Eliminate the printing of each division result (line 180).

As a first step toward searching for the pair of numbers that gives the maximum number of steps, find the number of steps required for each pair of numbers N1, N2 between 1 and some BOUND in Exercises 3-6 and 3-7.

Exercise 3-6. Initialize BOUND. Insert line

```
105 BOUND = 10
```

(You can try a larger BOUND once the program is working.)

Exercise 3-7. We want the computer to generate the numbers N1 and N2 to be tested, so delete lines 120, 130, and 140, which input N1 and N2.

Exercise 3-8. Use nested FOR-NEXT loops to run through all the pairs N1, N2. Insert lines

```
120 FOR N1 = 1 TO BOUND
130 FOR N2 = 1 TO BOUND
270 COUNTER = 0
280 NEXT N2
290 NEXT N1
```

Exercise 3-9. Run the program in Exercise 3-8.

Exercise 3-10. If you can't read as fast as the computer prints, make the program pause after each printing.

Exercise 3-11. Run this pausing version. Be sure it is giving the correct number of steps for each pair before going on.

Exercise 3-12. This program considers most pairs of numbers twice—once with N1 as the smaller number and once with N1 as the larger. Modify the FOR-NEXT loops so that each pair is considered only once.

Exercise 3-13. What pair of numbers between 1 and 10 gives the largest number of steps? Between 1 and 20? Between 1 and 30?

This current program finds the GCD and the number of steps for each pair of numbers, but we must scan the results to find the pair with the maximum number of steps. Now we upgrade the program so that the computer reports the maximum. We use a variable MAX to record the largest number of steps observed so far and print the GCD and number of steps each time a new maximum is obtained. Otherwise, we skip the printing.

The current lines 190 and 250 are

```
190 IF R = 0 THEN 250:REM. . .
250 PRINT:PRINT "GCD. . .
```

Now we want to skip the printing if the number of steps (COUNTER) does not exceed the current maximum (MAX).

Exercise 3-14. Change line 190 to

```
190 IF R = 0 THEN 245
```

and insert line

```
245 IF MAX >=3 COUNTER THEN 270
```

If MAX < COUNTER, the pair and number of steps will be printed. However, MAX must be updated before going on to a new pair.

Exercise 3-15. Insert line

```
267 MAX = COUNTER
```

Exercise 3-16. Raise BOUND to 100, and run the program.

Exercise 3-17. Make a conjecture. Prove it.

Exercise 3-18. Modify the program so that it prints each pair whose number of steps equals the current maximum. Note how the pairs change. Can you find a pattern?

4

DIVISORS

Many interesting properties of numbers involve their divisors. For example, 6 = 2*3, so 6 is ‘‘produced’’ by 2 and 3. Also, 10 = 5*2, so 5 produces 10. With respect to this producing idea, the number 7 stands out among the numbers 1 to 10. No other number produces 7 (the trivial factoring 7 = 1*7 doesn’t count), and 7 is not involved in producing any of the other numbers from 1 to 10. For this reason, the number 7 has been considered special. (In the biblical account of creation, for example, God rested on the seventh day.) In this chapter and the next, we investigate some properties of numbers having to do with divisors.

The following program, DIVISORS, lists all divisors of a user-supplied number N.

LISTING 4-1

```
100  REM    DIVISORS
110  HOME
120  PRINT "FOR WHAT NUMBER DO YOU"
130  PRINT : PRINT "WANT THE DIVISORS";
140  INPUT N
150  PRINT : PRINT " THE DIVISORS OF ";N;" ARE"
160  PRINT
170 K = 1
180  IF  INT (N / K) <  > N / K THEN 200
190  PRINT K,
200 K = K + 1
210  IF K <  = N THEN 180
```

Line 180 uses the function INT to test whether K is a divisor of N. The use of INT in DIVISORS is based on the fact that INT(X) = X if and only if X is an integer. In line 180, if INT(N/K) is not equal to N/K, N/K is not an integer, so K is not a divisor of N.

Run DIVISORS for some small numbers to see that it works. Also try it on some large numbers (such as 45,678).

Exercise 4-1. Modify DIVISORS in order to increase its speed. Print both K and N/K when a divisor K is found. (But don't print any divisor twice!) Try this out on some large numbers. Is it faster than DIVISORS?

The divisors of a number are the parts that go together to make up the number. One way to measure the parts of a number is to compute the sum of its divisors.

Exercise 4-2. Modify the program of Exercise 4-1 to also print the sum $\sigma(N)$ of all divisors of N. Run this program for various N.

A number (other than 1) cannot be a factor of itself, so it cannot play a role in producing itself. For this reason, a number is not considered to be a *proper* divisor of itself.

Exercise 4-3. Modify the program of Exercise 4-2 to also print the sum s(N) of all proper divisors of N.

1. Run this version for various N.
2. Find a number N such that $s(N) > N$.
3. Find a number N such that $s(N) < N$.
4. Find a number N such that $s(N) = N$.

The proper divisors of a number N are the parts from which it is produced. If the sum of these divisors exactly equals the number N itself, N is said to be *perfect*. If N has too few divisors to be perfect, it is called *deficient*. If N has too many divisors, it is called *abundant*. In terms of the sum s(N), a number N is

perfect if $s(N) = N$
abundant if $s(N) > N$
deficient if $s(N) < N$

Exercise 4-4. Modify the program of Exercise 4-3 to also state whether N is perfect, abundant, or deficient.

Exercise 4-5. Modify the program of Exercise 4-4 so that it tests each of the numbers from 1 to 30. How many of the numbers from 1 to 30 are abundant? perfect? deficient?

You see from the results of the previous exercise that 6 and 28 are perfect numbers. In the biblical account of creation, God created the universe in 6 days. Moreover, the moon's cycle, from one full moon to the next, takes 28 days! These appearances of perfect numbers and the fact that they are somewhat rare make finding them exciting. The next exercise challenges you to search for perfect numbers.

Exercise 4-6. Modify the program of Exercise 4-5 so that it prints just the perfect numbers. List all perfect numbers up to 10,000.

Even with the computer as an aid, searching for the perfect numbers by testing each and every number is very time-consuming. Imagine what it would be like to do the search by hand! It might speed the search to look for patterns or properties in the perfect numbers you have found and then restrict the search to numbers of that special form. Do the perfect numbers you have found share any special properties?

The idea of using properties of the perfect numbers is pursued in the next chapter. For now, change the question a bit by extending the idea of perfect numbers:

$\sigma(N) = 2N$ means N is perfect.
$\sigma(N) = 3N$ means N is triply perfect.
$\sigma(N) = 4N$ means N is quadruply perfect.

and so forth.

Exercise 4-7. Modify the program of Exercise 4-6 to search for such numbers.

Exercise 4-8. Are most numbers abundant? deficient? Compute the proportion R(N) of abundant to deficient numbers up to N, for $N = 1, 2, 3, \ldots$. Is R(N) a familiar function? Does R(N) increase or decrease as N increases?

5

PRIMES

In the Chapter 4, ''Divisors,'' we looked at the divisors of a number as the parts from which it was constructed. Among the divisors of a number, the prime divisors are the most fundamental. A *prime* number P is a whole number larger than 1, such that P and 1 are its only divisors. For example, 2, 7, and 11 are prime, but 4, 10, and 15 are not.

In this chapter, we study ways to construct tables of primes and use the tables to find the prime factors of numbers. We also use factorization into primes to generate a more efficient search for perfect numbers.

The sieve of Eratosthenes is a standard method for constructing a table of all the prime numbers up to a given number BOUND. The procedure, which we will use in two programs, is as follows: Write down all the numbers from 1 to BOUND. Cross off 1, because 1 is not prime (by definition). Then 2 is prime. Cross off all higher multiples of 2 by starting at 2 and crossing off every second number. Then 3 is the first number beyond 2 which has not been crossed off, so 3 is prime. Now cross off higher multiples of 3 by starting with 3 and crossing off every third number. In general, the deleting of multiples of a prime N is done by counting N steps from one multiple to the next. When all the higher multiples of N have been deleted, the first number above N which has not been crossed off is the next prime. Then the process of crossing off multiples is repeated with this new prime.

The program SIEVE DEMO (see Listing 5-1) gives a visual demonstration of this sieve method.

LISTING 5-1

```
100  REM      SIEVE DEMO
110  HOME
120  PRINT "    THIS PROGRAM DEMONSTRATES THE SIEVE"
130  PRINT : PRINT "METHOD FOR CONSTRUCTING A TABLE OF"
140  PRINT : PRINT "PRIMES."
150  VTAB (23)
160  PRINT "PRESS ANY KEY TO CONTINUE. "
170  GET ANS$
180  HOME
190 BOUND = 220: REM  SIEVE UP TO 220
200  REM   INITIALIZE ARRAY:
210  DIM A(BOUND)
220  FOR K = 1 TO BOUND
230 A(K) = K
240  NEXT
250  REM          PRINT TABLE:
260  REM
270  FOR N = 1 TO BOUND
280 NUM = A(N)
290  GOSUB 870: REM     PLACE NUM ON SCREEN
300  NEXT
310  REM  BEGIN SIEVE
320 A(1) = 0: REM  1 NOT PRIME
330  REM  CROSS 1 OFF SCREEN:
340  VTAB (1): HTAB (1): PRINT " . ";
350 N = 2
360  GOTO 510: REM  FLASH 2
370  HTAB (1): VTAB (24): PRINT A(N);" IS PRIME. PRESS ANY KEY
 TO GO ON";
380  GET ANS$: REM     PAUSE
390  REM  CLEAR TEXT LINE:
400  HTAB (1): VTAB (24): PRINT  SPC( 39);
410 K = 2: REM    START SIEVING WITH 2*N
420  IF K * N > BOUND THEN 470: REM  DONE WITH MULTIPLES OF
N
430  GOSUB 670: REM     REMOVE K*N FROM SCREEN
440 A(K * N) = 0: REM  REMOVE K*N FROM ARRAY
450 K = K + 1: REM  NEXT K
460  GOTO 420
470 NUM = A(N): GOSUB 870: REM   STOP FLASHING NUM
480 N = N + 1
490  IF N > BOUND THEN 530: REM    END
500  REM    IF A(N) <> 0, N IS NEW PRIME, SO FLASH IT:
510  IF A(N) <  > 0 THEN  FLASH :NUM = A(N): GOSUB 870: NORMAL
: GOTO 370
520  GOTO 480: REM  NEXT N
530  HTAB (2): VTAB (23)
540  PRINT "ALL DONE!";
550  END
560  REM
570  REM    SBRTN TO FIND SCREEN LOCATION FOR NUM
580  REM    VRTL IS VERTICAL TAB
590  REM    HRZL IS HORIZONTAL TAB
600  REM
610 QUOT =  INT ((NUM - 1) / 10)
620 RDR = (NUM - 1) - QUOT * 10
630 VRTL = QUOT + 1
640 HRZL = 4 * RDR + 1
650  RETURN
660  REM
670  REM    SBRTN TO REMOVE K*N FROM SCREEN
680  REM
690  REM  IF K*N ALREADY REMOVED, SHOW BRIEFLY IN INVERSE
```

```
700  REM   IF K*N A NEW REMOVAL, FLASH BEFORE REMOVING
710 NUM = K * N
720  IF A(K * N) = 0 THEN  INVERSE
730  IF A(K * N) <  > 0 THEN  FLASH
740  GOSUB 870: REM  PRINT K*N ON SCREEN
750  IF N = 2 THEN 800: REM  SKIP PAUSE
760  IF A(K * N) <  > 0 THEN  PRINT "";: REM  RING BELL
770  REM  PAUSE:
780  FOR J = 1 TO 500
790  NEXT
800  NORMAL
810  REM  NOW BLACK OUT K*N:
820  VTAB (VRTL)
830  HTAB (HRZL)
840  PRINT " . "
850  RETURN
860  REM
870  REM   SBRTN TO PLACE NUM ON SCREEN
880  REM
890  GOSUB 570: REM   FIND SCREEN PLACE FOR NUM

900  VTAB (VRTL)
910  REM  MOVE SMALLER NUMBERS TO RIGHT:
920  IF NUM <  = 9 THEN  HTAB (HRZL + 2)
930  IF 10 <  = NUM AND NUM <  = 99 THEN  HTAB (HRZL + 1)
940  IF NUM >  = 100 THEN  HTAB (HRZL)
950  PRINT NUM
960  RETURN
```

Exercise 5-1. Run SIEVE DEMO. Note that a number about to be deleted from the table is flashed. When the program looks at a number which has already been deleted, the number appears briefly in inverse mode.

Exercise 5-2. Run SIEVE DEMO and sieve on 2, 3, and 5. Now 7 is known to be prime. Before starting to sieve on 7, look at the multiples of 7 that remain in the table. Note that 49 is the first multiple of 7 to get crossed off. Now start the sieving on 7, and observe that the program checks 14, 21, 28, 35, and 42 before getting to 49. This is a significant waste of time.

How can we improve the program to avoid this unnecessary checking? When the program is ready to delete the multiples K*N of a prime N, the program begins (in line 410) with K = 2, whereas we know that N*N is the first multiple still appearing in the table. Change line 410 so that the sieving begins with N*N. Run this version. Note the increased speed of this version compared with the previous one.

Although this version is faster, it can still be improved. Run it again, and note that, after a while, the program finds no multiple to cross off for the new prime, but it keeps checking anyway. This is because the current version of the program continues to look in the table for multiples of a prime N as long as N ≤ BOUND (see line 490). But we know that N*N is the first multiple of N that can possibly be in the table. So, if N*N is too large to be in the table, all the remaining numbers must be prime and the program may as well end.

Exercise 5-3. Change line 490 as suggested above. Run this version. Note the decrease in the time required to complete the table.

In Exercises 5-2 and 5-3, you have used mathematics to make significant improvements in the sieving process. These improvements are incorporated in the program SIEVE, which will be the basis for some more work with primes. The program SIEVE constructs a table of primes without all the fancy printing and puts the primes in an array that may then be used in other computations.

To see the overall structure of the program, examine the flow chart for SIEVE in Figure 5-1. Now let's analyze the program SIEVE. Here is a listing of SIEVE.

LISTING 5-2

```
100  REM     SIEVE
110  REM
120  REM   PROGRAM TO COMPUTE PRIME TABLE
130  REM    BY SIEVE METHOD, WITH IN-PROCESS COMMENTS
140  REM
150  REM
160  REM    TO SIEVE TO AN ARRAY A(N).  IF N IS PRIME,
170  REM   A(N)=N, OTHERWISE A(N)=0
180  REM
190  HOME
200  PRINT "UP TO WHAT BOUND DO YOU WANT THE PRIMES";
210  INPUT BOUND
220  PRINT : PRINT "PRINT PRIME TABLE? (Y OR N)";
230  INPUT ANS$
240  DIM A(BOUND)
250  HOME
260  HTAB (13): VTAB (10)
270  PRINT "INITIALIZING ARRAY"
280  FOR K = 1 TO BOUND
290 A(K) = K
300  NEXT
310  HOME
320  REM    BEGIN SIEVE
330 A(1) = 0: REM  1 IS NOT PRIME
340 N = 2: REM  START DELETING MULTIPLES OF 2
350 K = N: REM  START DELETING WITH N*N
360  PRINT "SIEVING ON ";N
370  IF K * N > BOUND THEN 400: REM  FINISHED WITH MULTIPLES
OF N
380  REM   CROSS OUT K*N
390 A(K * N) = 0:K = K + 1: GOTO 370
400 N = N + 1: REM   NEXT N
410  IF N * N > BOUND THEN 460: REM  STOP SIEVING
420  IF A(N) <  > 0 THEN 350: REM  N IS PRIME, START SIEVING
ON N
430  HTAB 20: PRINT N;" IS OUT"
440  REM  A(N)=0 MEANS N ALREADY CROSSED OUT
450  GOTO 400: REM   GO TO NEXT N
460  PRINT : PRINT "SIEVE COMPLETE": PRINT
470  REM
480  REM
490  REM   SUBROUTINE TO MAKE ARRAY
500  REM   OF PRIMES PM(K)
510  PRINT "COUNTING PRIMES": PRINT
520 N = 0
530  FOR K = 1 TO BOUND
540  IF A(K) > 0 THEN N = N + 1
550  NEXT
560 B2 = N: REM  N PRIMES WERE FOUND
570  PRINT B2;" PRIMES WERE FOUND UP TO ";BOUND: PRINT
```

```
580  PRINT "LOADING PRIME TABLE PM(K)"
590  DIM PM(B2)
600 N = 1
610  FOR K = 1 TO BOUND
620  IF A(K) <  > 0 THEN PM(N) = A(K):N = N + 1
630  NEXT
640  REM
650  PRINT : PRINT "PRIME TABLE COMPLETE AS PM(K),"
660  PRINT : PRINT "FOR 1 <= K <= ";B2
670  PRINT
680  IF ANS$ <  > "Y" THEN 720
690  FOR K = 1 TO B2
700  PRINT PM(K),
710  NEXT
720  END
```

In lines 280 to 300, all the numbers from 1 to BOUND are "written down" in an array A(K).

The sieving is done in lines 320 to 460.

At line 390, a multiple of N is "crossed off" by setting the array entry A(K*N) to zero.

The primes are counted in lines 510 to 570.

The primes are "written down" in a table in lines 580 to 710.

Exercise 5-4. Run SIEVE for various sizes of tables. Note how the time required to sieve on a particular prime N decreases with larger N.

Every whole number can be factored into primes. For example, 24 = 2*2*2*3, and 175 = 5*5*7. To find the prime factorization of a number N, you may use the table of primes constructed by SIEVE. Work up through the table, and test each prime P to see if it is a factor of N. If P is not a factor, move up to the next prime. If you reach a prime P such that P*P > N, then N itself is prime, and the process stops. If you find a prime P that is a factor, print P, and then continue with the other factor, N/P.

Exercise 5-5. Add lines at the end of the program SIEVE to create a program that will compute the prime factorization of an input number N. Use the flow chart in Figure 5-2.

Exercise 5-6. Compute and record the prime factorization of

1. 45,678, 56,789, and 34,567
2. 3,646,313 and 3,573,349
3. 170,500 and 645,150
4. 1,055,449 and 1,714,781

One use for prime factorization is in finding greatest common divisors. The GCD of two numbers is equal to the product of their common prime factors. For example, 315 = 3*3*5*7 and 330 = 2*3*5*11, so their common prime factors are 3 and 5. Hence, their GCD is 3*5 = 15.

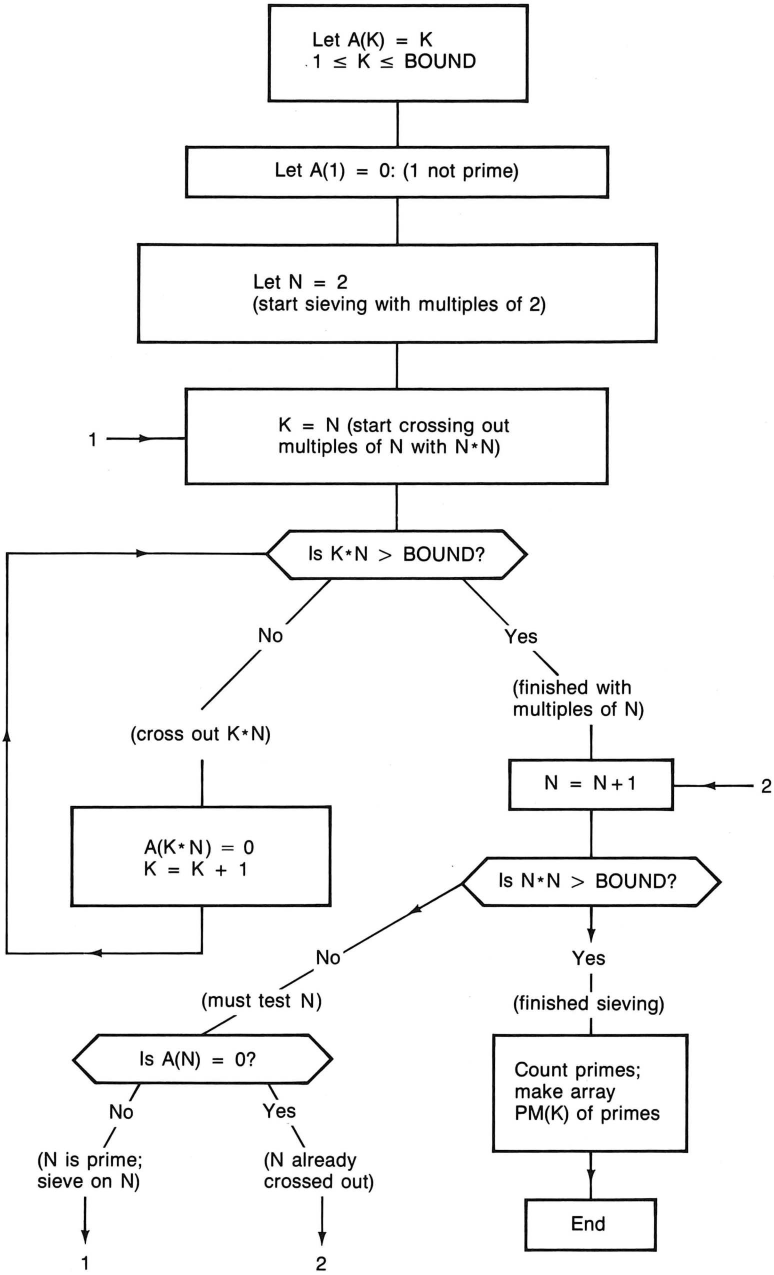

Let A(K) = K
1 ≤ K ≤ BOUND
Let A(1) = 0: (1 not prime)
Let N = 2
(start sieving with multiples of 2)
1
K = N (start crossing out
multiples of N with N*N)
Is K*N > BOUND?
No
Yes
(cross out K*N)
(finished with
multiples of N)
A(K*N) = 0
K = K + 1
N = N+1
2
Is N*N > BOUND?
No
(must test N)
Yes
(finished sieving)
Is A(N) = 0?
No
Yes
(N is prime;
sieve on N)
(N already
crossed out)
Count primes;
make array
PM(K) of primes
End
1
2

Figure 5-1 Flow chart for SIEVE

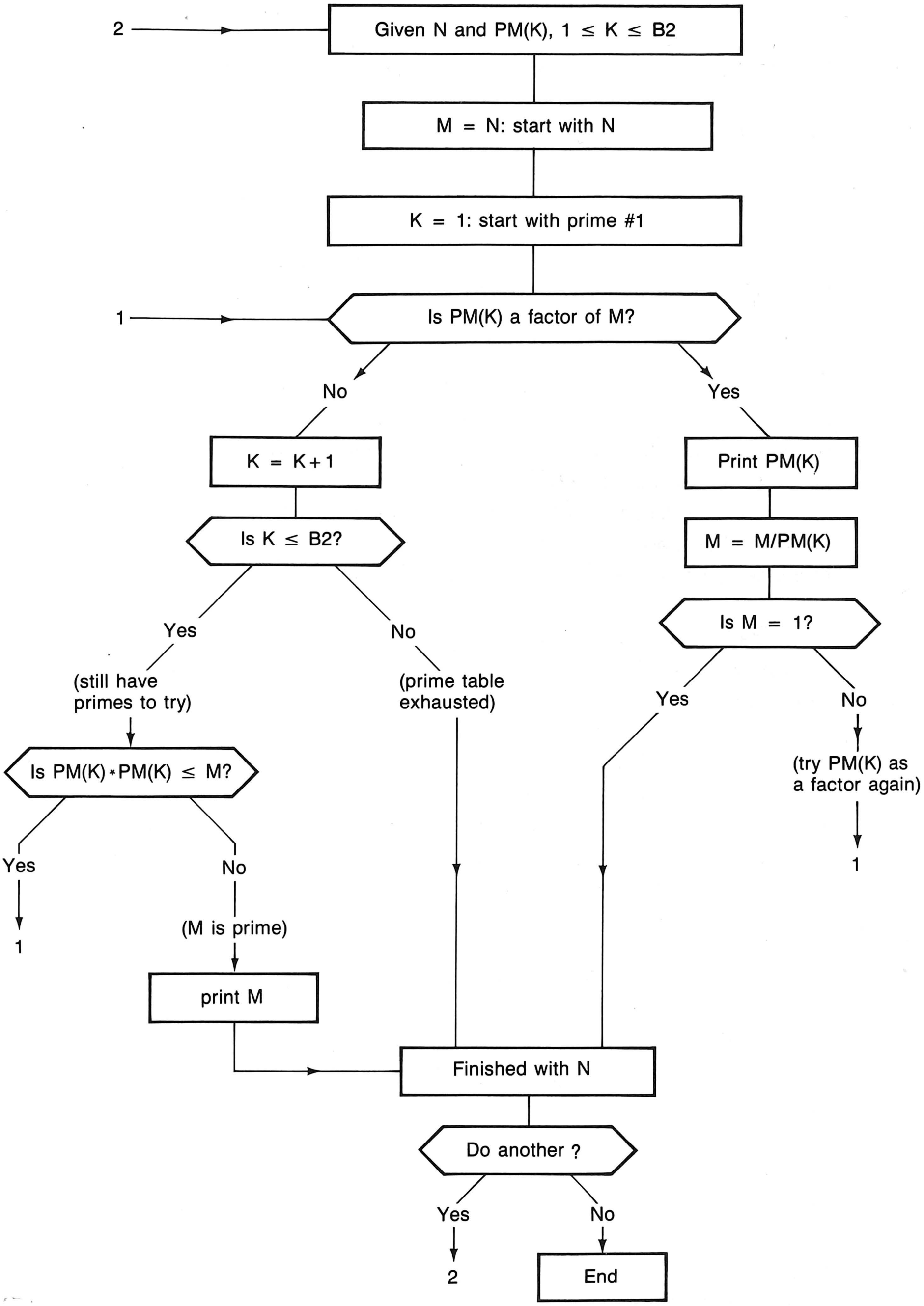

2
Given N and PM(K), 1 ≤ K ≤ B2
M = N: start with N
K = 1: start with prime #1
1
Is PM(K) a factor of M?
No
Yes
K = K+1
Print PM(K)
Is K ≤ B2?
M = M/PM(K)
Is M = 1?
Yes
No
(still have primes to try)
(prime table exhausted)
Yes
No
(try PM(K) as a factor again)
Is PM(K)*PM(K) ≤ M?
1
Yes
No
1
(M is prime)
print M
Finished with N
Do another ?
Yes
No
2
End

Figure 5-2 Flow chart for prime factorization

If a prime is repeated in each number, it appears the smaller number of times as a factor in the GCD. For example, 84 = 2*2*3*7 and 120 = 2*2*2*3*5. The factor 2 is repeated in both numbers. It is repeated three times in 120, but only twice in 84, so 2*2 must be included in the GCD. The remaining common prime factor is 3, so the GCD is 2*2*3 = 12.

Exercise 5-7. Use the prime factorizations you found for the pairs in Exercise 5-6, parts 2, 3, and 4, to find the GCD of each pair.

Now that you have used prime factorization to compute GCDs, compare this method to the Euclidean algorithm method which we met earlier in the program GCD DEMO.

Exercise 5-8. Use the program GCD DEMO to compute the GCD of each of the pairs in Exercise 5-6, items 2, 3, and 4. Compare the results with the GCDs you found from the prime factorizations. Do the two methods give the same answers? Which method is faster in getting the GCD?

We now return to the problem of constructing tables of prime numbers. We used the sieve method above, but you could also just test each number for primeness and insert those which pass the test in an array.

Exercise 5-9. Write a program to compute a table of primes PM(K) by testing each number and inserting the primes in the array. Use as efficient a method as you can. Compare the speed of your program to that of SIEVE.

Exercise 5-10. Make SIEVE more efficient. (For one improvement, start with an array of zeros and insert the nonprimes, rather than loading an array with the numbers 1 to BOUND and then crossing out nonprimes.)

Is sieving to get primes as efficient as the direct method used in Exercise 5-9?

Here are some opportunities for investigating other questions about primes by using the prime tables.

Twin primes are those which differ by 2 (which is as close as most primes can get to each other). For example, 11 and 13 are twin primes.

Exercise 5-11. List as many twin prime pairs as you can.

It is not known whether there is an infinite number of prime twins. The gaps between successive primes can become quite large. (For example, 100! + 2, 100! + 3, . . . , 100! + 100 is a sequence of 99 consecutive nonprime numbers, because 100! + K is divisible by K.)

Exercise 5-12. Find the longest gap between successive primes in your prime table.

Although the gaps between successive primes may seem random, so that the primes get sparse in a random fashion, the proportion of primes to nonprimes shows some regularity.

Exercise 5-13. For N = 1, 2, . . . , ?, compute PI(N) = number of primes $\leq$ N and the proportion PI(N)/N.

Exercise 5-14. A famous result of number theory, the prime number theorem, says that PI(N)/N behaves like 1/log(N) in the sense that the quotient

$$\frac{\mathrm{PI(N)/N}}{1/\log(\mathrm{N})}$$

is close to 1 for large N. (The logarithm here is the natural, base *e*, logarithm, which is given by LOG(X) in BASIC.) To test out this theorem, compute the quotient above for N = 1, 2, . . . , ?.

6

PERFECT NUMBERS AGAIN

We now briefly return to the search for perfect numbers. You have seen in Exercise 4-6 that 6, 28, and 496 are the first three perfect numbers. You also saw that to search for perfect numbers by checking each number is very slow. We now speed up the search by using information provided by prime factorization.

Exercise 6-1. Use your prime factorization program from the previous chapter to compute and record the prime factorizations of 6, 28, and 496. Note that their factorizations all have the same form. Could it be that perfect numbers all have this form? This may be too much to hope for, but you might find perfect numbers more quickly by testing only numbers of this form.

The program PERFECT SEARCH (see Listing 6-1) uses the naive test-every-number approach to finding perfect numbers.

LISTING 6-1

```
100  REM     PERFECT SEARCH
110  HOME
120  PRINT "      THIS PROGRAM SEARCHES FOR"
130  PRINT : PRINT "PERFECT NUMBERS BY TESTING ALL NUMBERS"
140  PRINT : PRINT "FROM 1 UP TO M."
150  PRINT : PRINT "      UP TO WHAT M DO YOU WANT TO SEARCH";:
 INPUT M
160  HOME
170  PRINT "      OF THE NUMBERS UP TO ";M;","
180  PRINT : PRINT
190  FOR N = 1 TO M
200  REM  K IS POSSIBLE DIVISOR
210  REM  S IS SUM OF DIVISORS
```

```
220  REM  INITIALIZE K AND S:
230 K = 1
240 S = 0
250  IF  INT (N / K) <  > N / K THEN 290: REM  K NOT A DIVISOR
, SO GO TO NEXT K
260  REM  HERE, K IS A DIVISOR, SO ADD K AND N/K TO S
270 S = S + K
280  IF N / K <  > K THEN S = S + N / K
290 K = K + 1
300  IF K * K <  = N THEN 250
310  REM  NOW HAVE S THE SUM OF ALL DIVISORS
320  IF S - N = N THEN  PRINT N;" IS PERFECT": PRINT
330  NEXT : REM  GO TO NEXT N
340  PRINT "ALL DONE"
```

Run PERFECT SEARCH to see that it works and is *slow*.

Exercise 6-2. Modify PERFECT SEARCH so that it tests only those numbers of the form you found in Exercise 6-1. Use this program to find as many perfect numbers as you can. Record the perfect numbers you find.

Exercise 6-3. Compute and record the prime factorizations for each of the perfect numbers found in Exercise 6-2. Now reexamine the form of the numbers. Note that the number is perfect only when it has just one prime factor other than 2. You can use this fact (if you haven't already) to speed up the search even more.

The numbers of the form we have been considering are all even, so we have ignored the possibility that an odd number might be perfect. You might like to search for an odd perfect number. If you find one, you'll be famous! As of now, no one knows whether any odd perfect numbers exist.

Another question about perfect numbers is whether there is a largest one. They are so scarce, perhaps there is only a finite number of them. No one knows the answer to this one either. In contrast to the first question, this is a question which you cannot answer with a computer! That is, unless you have a computer capable of handling arbitrarily large numbers and you have an infinite amount of time to check *every* number.

7

REDUCING FRACTIONS

When doing arithmetic, most of us would rather work with decimals than fractions. This is because the procedures for adding and multiplying fractions are complicated and the reducing of fractions to lowest terms is tedious. On the other hand, fraction arithmetic avoids the round-off error that is inherent in decimal arithmetic. We see this later, for example, in solving linear equations in Chapters 17 through 20. Fraction arithmetic also reveals some mathematical facts that would be obscured by decimals, as we will see in studying means and Farey sequences. This chapter and the next develop a program to make the computer take over the drudgery of fraction arithmetic.

We want the results of fraction operations to be given in lowest terms, so we first develop a program to reduce a fraction. In order to reduce a fraction to lowest terms, we must compute the greatest common divisor (GCD) of the numerator and denominator. For example, to reduce 16/24:

$$\text{GCD}\ (16,\ 24) = 8$$

and 16 and 24 factor as $16 = 2*8$ and $24 = 3*8$, respectively. Hence, dividing the numerator 16 and the denominator 24 by the GCD 8 gives the reduced form $\frac{2}{3}$. The program GCD DEMO computes greatest common divisors for pairs of whole numbers, so we use it in reducing fractions.

Our program must be able to reduce fractions with negative numerators or denominators, and we want the reduced form to have a positive denominator. For example, $(-16)/24$ and $16/(-24)$ should reduce to $(-2)/3$, and $(-16)/(-24)$ to $\frac{2}{3}$. Thus we need to see what GCD DEMO does for negative integers.

Exercise 7-1. Run GCD DEMO for the pairs (6, −4), (−4, 6), (−4, −6), and (−6, −4). Note that it does not give the *greatest* common divisor in all these cases. Its answer is, however, exactly the correct common factor we must remove to reduce a fraction to one with a positive denominator! Was Euclid looking ahead to fractions, or are we just lucky?

Our goal is a program that does arithmetic with fractions, either multiplication or addition, at the discretion of the user. The program has to do various operations: multiply two fractions, add two fractions, and reduce a fraction (and hence, find a GCD). We will build the program out of subroutines, one for each of these operations.

The following program, REDUCE, illustrates the use of a subroutine. It uses the Euclidean algorithm part of the GCD DEMO as a subroutine in order to reduce a fraction.

LISTING 7-1

```
100  REM     REDUCE
110  REM  PROGRAM TO REDUCE FRACTION
120  REM  USING GCD SUBROUTINE
130  REM
140  HOME
150  PRINT "WHAT FRACTION DO YOU WANT TO REDUCE?"
160  PRINT : PRINT "NUMERATOR = ";: INPUT NUM
170  PRINT : PRINT "DENOMINATOR = ";: INPUT DEN
180  REM   SET UP FOR GCD SUBROUTINE:
190 A = NUM:B = DEN
200  GOSUB 1000: REM   GCD SUBROUTINE
210  REM  SUBROUTINE RETURNS GCD
220  REM   NOW DIVIDE OUT BY GCD:
230 NNUM = NUM / GCD:NDEN = DEN / GCD
240  PRINT : PRINT "REDUCED FORM OF ";NUM;"/";DEN;" IS ";
250  PRINT NNUM;"/";NDEN
260  END
1000  REM  GCD SUBROUTINE:A,B IN,GCD OUT
1010  REM  SUBROUTINE ASSUMES B<>0
1020 Q =  INT (A / B): REM  DIVIDE A BY B
1030 R = A - Q * B: REM  REMAINDER
1040  IF R = 0 THEN 1080: REM   ALGORITHM FINISHED,GCD IS B
1050  REM  IF R <> 0 MUST DO ANOTHER DIVISION
1060  REM  NOW SET UP FOR NEXT DIVISION
1070 A = B:B = R: GOTO 1020
1080 GCD = B
1090  RETURN
```

The main program is in lines 100 to 260.

Lines 150 to 170 obtain the fraction to be reduced.

Lines 180 and 190 set up the variables A and B for the GCD subroutine.

Line 200 calls the GCD subroutine.

At the completion of the subroutine, the program returns to line 210 and continues from there.

The GCD subroutine is in lines 1000 to 1090. The subroutine applies the Euclidean algorithm to the A and B supplied by the main program. Then in line 1080, it records the result of the algorithm in the variable GCD. In line 1090, it returns control to the main program.

When the program returns from the subroutine, line 230 divides the numerator and denominator by the result GCD of the subroutine. Finally, the original and reduced forms are printed by lines 240 and 250.

Exercise 7-2. Run REDUCE for all possible combinations of positive, negative, and zero numerators and denominators. Check that it gives the correct reduced form.

As you can see from running REDUCE, it could stand some improvement. The following exercises suggest ways of changing the program.

Exercise 7-3. Modify REDUCE so that it rejects zero denominators input in line 170.

Exercise 7-4. Modify REDUCE so that, if the operator types in 0 as a numerator, it skips the GCD subroutine.

Exercise 7-5. Improve the printing of the reduced form: If NDEN = 1, print just NNUM.

Exercise 7-6. Modify REDUCE so that the operator is given the option of reducing another fraction before the program ends.

Exercise 7-7. Modify REDUCE so that improper fractions are printed as mixed numbers.

8

FRACTION ARITHMETIC

In this chapter, we finish the development of a program to do arithmetic with fractions, which we began in Chapter 7. We use this fraction arithmetic program later in solving problems that demand fraction solutions. The program must be able to add and multiply fractions and give the answers in reduced form. We use subroutines to do the adding and multiplying. Each of these subroutines makes use of a reducing subroutine that converts the answer to lowest terms.

The (incomplete) program FRAC ARITH forms the foundation for the arithmetic program. It contains some of the necessary subroutines, for example, the reducing and GCD subroutines of Chapter 7. You will add to FRAC ARITH to construct the complete fraction arithmetic program.

Listing 8-1 shows the edited forms of the programs GCD and REDUCE as subroutines in FRAC ARITH.

LISTING 8-1

```
670  REM  GCD SUBROUTINE:A,B IN,GCD OUT
680  REM  SUBROUTINE ASSUMES B<>0
690 Q =  INT (A / B): REM  DIVIDE A BY B
700 R = A - Q * B: REM  REMAINDER
710  IF R = 0 THEN 750: REM   ALGORITHM FINISHED,GCD IS B
720  REM  IF R <> 0 MUST DO ANOTHER DIVISION
730  REM  NOW SET UP FOR NEXT DIVISION
740 A = B:B = R: GOTO 690
750 GCD = B
760  RETURN
780  REM
790  REM    SUBROUTINE "REDUCE" TO REDUCE FRACTION
```

```
800  REM  USING GCD SUBROUTINE
810  REM    NUM, DEN IN, NNUM,NDEN OUT
820  REM  IF NUM = 0, SKIP GCD SUBRTN
830  IF NUM = 0 THEN NNUM = 0:NDEN = 1: RETURN
840 A = NUM:B = DEN
850  GOSUB 690: REM  GCD SUBROUTINE
860  REM  SUBROUTINE RETURNS GCD
870  REM   NOW DIVIDE OUT BY GCD:
880 NNUM = NUM / GCD:NDEN = DEN / GCD
890  REM  REDUCED FORM IS NNUM/NDEN
900  RETURN
```

Note that the reducing subroutine (lines 790 to 900) expects a fraction NUM/DEN and returns the reduced form as NNUM/NDEN. If the original numerator NUM is zero, it skips the GCD subroutine and immediately gives the reduced form as $^{0}/_{1}$ (see lines 820 and 830). Otherwise, it calls the GCD subroutine in line 850, divides out by the common factor in line 880, and then returns to the main program.

The GCD subroutine, lines 670 to 760, does not protect itself against zero denominators, so the main program must be careful not to allow a zero denominator to reach this subroutine.

The program REDUCE of Chapter 7 inputs fractions by asking separately for the numerator and denominator. This gets very tedious, particularly when we have to enter several fractions, as we will when solving systems of equations. Hence, we want to allow a fraction to be entered in the form X/Y. We can do this by having X/Y input as a string variable and then extracting X and Y by searching through the string for the "/".

The subroutine GETTER (see Listing 8-2) does the extracting of the numerator and denominator from a string of the form X/Y. We use this subroutine in future programs that require fractions to be entered from the keyboard.

LISTING 8-2

```
1270  REM    SUBROUTINE "GETTER" TO READ FRAC FROM KEYBOARD

1280  REM   SUBRTN EXPECTS STRING OF FORM X/Y
1290  REM          AND EXTRACTS NUM=X AND DEN=Y
1300  REM   BEFORE ENTERING SUBROUTINE, SET
1310  REM    VV$= SOME CONNECTING PHRASE SUCH AS "A",OR "THE
FIRST",ETC.
1320  PRINT : PRINT "PLEASE TYPE ";VV$;"FRACTION"
1330  INPUT A$
1340  REM  SEARCH FOR "/" IN A$:
1350  FOR K = 1 TO  LEN (A$)
1360  REM  LOOK AT K TH CHARACTER IN A$:
1370 CHAR$ =  MID$ (A$,K,1)
1380  IF CHAR$ = "/" THEN 1450: REM   HAVE FOUND "/"
1390  NEXT : REM    HAVEN'T FOUND "/" SO ADVANCE K
1400  REM   HERE, A$ HAS NO "/";ASSUME A$ IS AN INTEGER
1410 NUM =  VAL (A$): REM  NUMERICAL VAL OF A$
1420 DEN = 1
1430  RETURN
1440  REM   HERE, HAVE FOUND "/" AS K-TH CHARACTER OF A$
1450 NUM$ =  LEFT$ (A$,K - 1)
1460 DEN$ =  RIGHT$ (A$, LEN (A$) - K)
1470 NUM =  VAL (NUM$)
1480 DEN =  VAL (DEN$)
1490  IF DEN = 0 THEN  PRINT : PRINT "DENOMINATOR NOT ALLOWED
TO BE ZERO.": GOTO 1320
1500  RETURN
```

Now let's analyze the subroutine GETTER.

Line 1330 obtains the fraction string A$.

The search for the "/" begins in line 1350. LEN(A$) is the number of characters in the string A$.

Line 1370 sets CHAR$ to the Kth character of A$. The subroutine ends when it finds the "/" or when it runs out of characters, which means that the string has no "/".

Line 1490 protects against zero denominators by sending careless users back to input another string.

Exercise 8-1. To see explicitly what the subroutine is doing, insert the following lines:

```
1355 PRINT "K = ";K
1375 PRINT "CHAR$ = "; CHAR$
1425 PRINT "NUM = "; NUM; " DEN = "; DEN
1485 PRINT "NUM = "; NUM; " DEN = "; DEN
```

Insert a pause line 1385.

Try this fraction reader subroutine on various fractions and integers by typing GOSUB 1270. You may get a mysterious error message at the end, because of this keyboard GOSUB command, but it's nothing to worry about; the subroutine will normally be called from the main program, not from the keyboard. Note that the subroutine does indeed extract the correct numerator and denominator.

Exercise 8-2. Return the subroutine to its original state: Delete the extra lines of Exercise 8-1.

FRAC ARITH contains a subroutine MULTIPLY, which multiplies two given fractions, and a subroutine MULTIPLY AND PRINT, which obtains two fractions from the keyboard, uses the MULTIPLY subroutine to compute their product, then prints out the result. Why do we split this operation into two subroutines? In later applications, for example, in solving linear equations, we want to multiply two fractions without printing the result. Hence we need the MULTIPLY subroutine to be separate.

We analyze these multiplying subroutines here, then you will use them as a guide to write similar subroutines for fraction addition.

LISTING 8-3

```
920  REM   SUBROUTINE "MULTIPY" TO MULTIPLY FRACS
930  REM  N1/D1 AND N2/D2 IN, N3/D3 OUT
940  REM
950  REM  COMPUTE UNREDUCED PRODUCT NUM/DEN
960 NUM = N1 * N2
970 DEN = D1 * D2
980  GOSUB 790: REM   REDUCE TO NNUM/NDEN
990  REM  SET UP FOR RETURN
1000 N3 = NNUM:D3 = NDEN
1010  RETURN
```

The subroutine MULTIPLY (see Listing 8-3) expects the fractions to be N1/D1 and N2/D2. In lines 960 and 970, it computes the unreduced product

```
NUM/DEN = (N1*N2) / (D1*D2)
```

Line 980 calls the subroutine REDUCE, which returns the reduced form NNUM/NDEN.

Lines 1000 and 1010 return the reduced form of the product as N3/D3.

The subroutine MULTIPLY AND PRINT (see Listing 8-4) gets two fractions from the keyboard (using the subroutine GETTER), multiplies them (using the subroutine MULTIPLY), and prints their product.

LISTING 8-4

```
2450  REM      SUBROUTINE "MULTIPLY AND PRINT"
2460  PRINT "WHAT FRACTIONS DO YOU WISH TO MULTIPLY?"
2470 VV$ = "FIRST "
2480  GOSUB 1270: REM  GETTER, NUM,DEN RETURN
2490 N1 = NUM:D1 = DEN: REM  FOR MULT SUBROUTINE
2500 VV$ = "SECOND ": REM   FOR GETTER
2510  GOSUB 1270 REM  GETTER, NUM, DEN RETURN
2520 N2 = NUM:D2 = DEN: REM  FOR MULT SUBROUTINE
2530  GOSUB 920: REM   "MULTIPLY" SUBRTN, N3,D3 RETURN
2540  PRINT : PRINT "THE PRODUCT OF ";N1;"/";D1;" AND ";N2;"/"
;D2;" IS ";N3;
2550  IF D3 <  > 1 THEN  PRINT "/";D3
2560  PRINT
2570  RETURN
```

Line 2480 calls the subroutine GETTER to obtain the first fraction.

Line 2470 tells GETTER to use the word ‘‘first’’ in asking for a fraction.

Line 2490 sets N1/D1 equal to the fraction NUM/DEN that GETTER obtained.

Lines 2500 to 2520 repeat this process and obtain the second fraction as N2/D2.

Now that the fractions to be multiplied are established as N1/D1 and N2/D2, the MULTIPLY subroutine is called by line 2530.

Then lines 2540 and 2550 print the factors and their product.

Exercise 8-3. Try this subroutine on various fractions by typing GOSUB 2450. Check that it gives correct answers and can handle zero denominators.

Now it's your turn. The following exercises ask you to write subroutines to do fraction arithmetic by mimicking the fraction multiplying we have just discussed. In these exercises, do your programming with line numbers 3000 and above to avoid destroying the given parts of FRAC ARITH.

Exercise 8-4. Use the subroutine MULTIPLY of lines 920 to 1010 as a guide, and write a subroutine ADD to add two fractions.

Exercise 8-5. Use the subroutine MULTIPLY AND PRINT of lines 2450 to 2570 as a guide, and write a subroutine to get two fractions from the keyboard, add them, and print their sum.

Exercise 8-6. Use the command GOSUB to test your subroutines as you did earlier for the multiplication subroutines. Be sure your subroutines give the correct sums.

To complete the fraction arithmetic program, we need a driver program that will present the user with a menu of choices: adding, multiplying, or stopping. Depending on the choice of the user, the driver will either call the appropriate subroutine or end.

Exercise 8-7. Write a driver program as described above. Insert a GO TO line as the first line of the entire program to transfer control to the driver.

9

MEANS

We are all accustomed to summarizing a sequence of numbers by its average. For example, if we know that the average score was 85 on a 100-point exam, we feel we have some overall information about the students' performance on the exam. This average, which we use so often, is just one of several means that can be used to summarize data. Here we will use the term *arithmetic mean* for the familiar average to distinguish it from the others we discuss—the *geometric mean* and the *harmonic mean*. We will see that certain kinds of practical problems demand the harmonic and geometric means for their solutions. In addition, the geometric and harmonic means are used later in approximating π.

Here are the definitions of the three means for a pair of numbers a and b.

Arithmetic Mean. The arithmetic mean of two numbers is

$$A(a, b) = \frac{a + b}{2}$$

Geometric Mean. The geometric mean of two nonnegative numbers is

$$G(a, b) = \sqrt{ab}$$

Harmonic Mean. The harmonic mean of two positive numbers is

$$H(a, b) = \frac{2ab}{a + b}$$

The harmonic mean is not as strange as it may seem at first. It is the reciprocal of the average of the reciprocals of a and b:

Reciprocals: $1/a$, $1/b$
Average: $(1/2)(1/a) + 1/b) = (1/2)[(a + b)/ab] = (a + b)/2ab$
Reciprocal: $2ab/(a + b) = H(a, b)$

We first see how each of these means can be constructed with ruler and compass. For the arithmetic mean, lay off segments of lengths a and b end to end on a line, starting at point A, and ending at point C. (See Figure 9-1.) Then the perpendicular bisector of segment AC cuts off a segment of length $x = A(a, b)$.

For the geometric mean, again lay off segments of lengths a and b end to end on a line. (See Figure 9-2.) Construct the semicircle on b + a as diameter. Then the semicircle cuts off $x = G(a, b)$ on the perpendicular at the junction point.

Exercise 9-1. Use similar triangles to prove that $x = \sqrt{ab}$.

The construction of the harmonic mean involves parallel lines. Lay off a segment of length 2a from point A to point B on a line. (See Figure 9-3.) From point A lay off segments of lengths a + b and b end to end on another line, ending at points C and D, respectively. Draw line CB. Then construct the line through D parallel to line CB. Let this parallel line cut line AB at point E. Then segment BE has length $x = H(a, b)$.

Exercise 9-2. Use similar triangles to prove that $x = 2ab/(a + b)$.

Exercise 9-3. Use a ruler and compass to construct the three means for a couple of segments. Compare the lengths of the mean segments with the original segments and with each other.

Exercise 9-4. Write a program to compute, for user-supplied numbers A and B, the

Arithmetic mean: (A + B)/2
Geometric mean: SQR(A*B)
Harmonic mean: 2*A*B/(A + B)

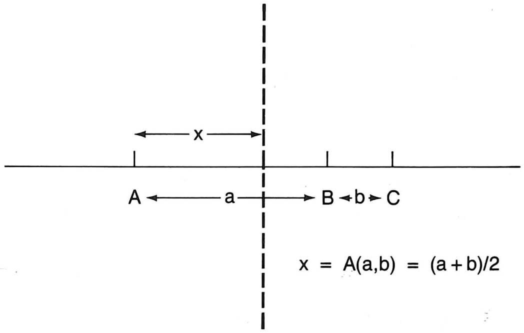
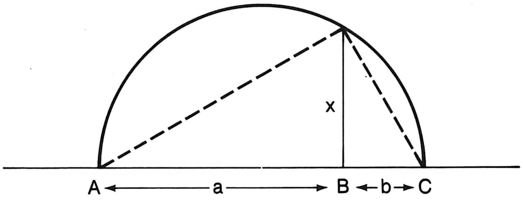


Figure 9-1 Arithmetic mean



$x = G(a,b) = \sqrt{ab}$

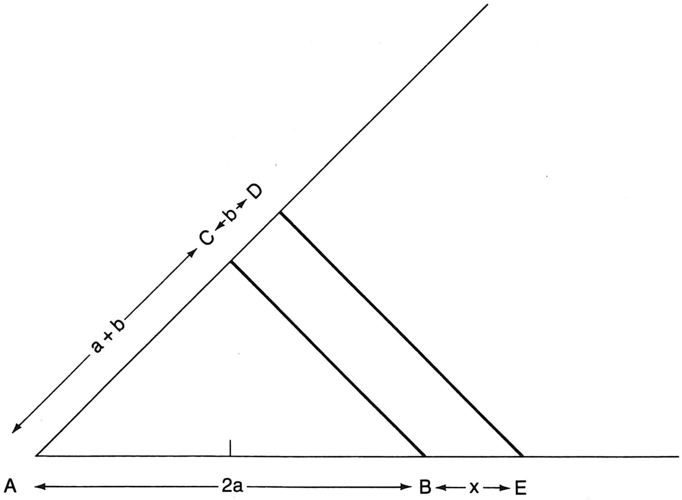
Figure 9-2 Geometric mean

Exercise 9-5. Compute the three means for various numbers A and B. Observe the numerical order of the means. Prove that the means always lie in this order.

The following examples show how each of these means arises in problems of equalization.

Example 9-1. Harry and Alphonse have two milk containers to carry to the truck. One contains 6 gallons, the other 4 gallons. If they distribute the milk equally between the two containers, how many gallons does each have to carry? The answer is the *arithmetic* mean: $(6 + 4)/2 = 5$ gallons.

Example 9-2. An oil company has a three-fold increase in its profits over a 5-year period, and then a two-fold increase over the next 9 years. Thus the profits increased by a factor of 6 over the 14 years. If P is the amount of profit at the start, then at



$x = H(a,b) = 2ab/(a+b)$

Figure 9-3 Harmonic mean

the end of the first period, they are P*2. At the end of the second period, they are (P*2)*3 = P*6.

As an industry analyst, you suspect that the profits are increasing by equal factors in equal times. If the six-fold increase in profits resulted from the same growth factor in each half of the 14-year period, what would that growth factor be? The answer is the *geometric* mean: $\sqrt{2*3}$ of the two growth factors 2 and 3. To see this, let x be the single growth factor. At the end of the first 7 years, the profits are P*x. At the end of the second 7 years, the profits are (P*x)*x = $P*x^2$. You want to find x so that $P*x^2 = P*6$. Hence, x must be $\sqrt{6}$.

Example 9-3. George drives to Buffalo at 30 mph and drives back at 70 mph. Alice wants to make the trip in the same amount of time but at a steady speed. At what speed should she drive? The answer is the *harmonic* mean—H(30,70) = 2*30*70/(30 + 70) = 42 mph.

To see this, let the distance to Buffalo be D. In general,

$$\text{distance} = \text{speed*time}$$

so George's trip to Buffalo at 30 mph took D⁄30 hours. His return trip at 70 mph took D⁄70 hours. So his total trip time was

$$\begin{aligned} T &= \frac{D}{30} + \frac{D}{70} \\ &= D\left(\frac{1}{30} + \frac{1}{70}\right) \\ &= \frac{D(70 + 30)}{30*70} \end{aligned}$$

hours. For Alice to use this same amount T of time for the total distance 2*D, she must travel at speed

$$\begin{aligned} x &= \frac{2D}{T} \\ &= 2D\left(\frac{1}{T}\right) \\ &= \frac{2D*30*70}{D(70 + 30)} \\ &= \frac{2*30*70}{70 + 30} \\ &= H(30,\ 70) \quad \text{mph} \end{aligned}$$

Another way to express the result of Example 3 is to say that the average speed of George's trip is the harmonic mean—not the arithmetic mean—of his two speeds.

You may use your program from Exercise 9-4 to compute the answers in the next three problems.

Exercise 9-6. Saudi Arabia plans to increase oil production by a factor of 1.5 in the next 5 years and then to double production in the following 5 years. Mexico wants to increase production by the same total factor over the next 10 years, but by equal factors in each 5-year period. By what factor should Mexico plan to increase production in each period?

Exercise 9-7. Susan and Mary must move two piles of bricks to the construction site. One pile contains 1,478 bricks, and the other one contains 2,990. If they share the work equally, how many bricks will each move?

Exercise 9-8. In a cross-desert auto race, Jerry and Hal will share the driving of their car. Jerry will drive the first half of the distance at 175 mph, and Hal will drive the second half at 160 mph. Bob and Louise will also share the driving of their car, each driving half the distance. To save wear and tear on the car, they each plan to drive at the same speed. In order to beat Jerry and Hal, how fast must they drive?

To make wider use of these means, we must be able to compute the means for sequences of numbers, not just for two numbers at a time as we did above. You already know how to do this for the arithmetic mean (the usual average). The geometric and harmonic means can also be generalized. Here are the definitions:

Arithmetic Mean. The arithmetic mean of n numbers $a_1, a_2, \ldots, a_n$ is

$$A(a_1, a_2, \ldots, a_n) = \frac{a_1 + a_2 + \ldots + a_n}{n}$$

Geometric Mean. The geometric mean of n nonnegative numbers $a_1, a_2, \ldots, a_n$ is

$$G(a_1, a_2, \ldots, a_n) = \sqrt[n]{a_1 * a_2 * \ldots * a_n}$$

Harmonic Mean. The harmonic mean of n positive numbers $a_1, a_2, \ldots, a_n$ is

$$H(a_1, a_2, \ldots, a_n) = \frac{1}{\frac{1}{n}\left(\frac{1}{a_1} + \frac{1}{a_2} + \ldots + \frac{1}{a_n}\right)}$$

Note that the harmonic mean is again the reciprocal of the average of the reciprocals.

The program MEANS (see Listing 9-1) computes the arithmetic, geometric, and harmonic means for user-supplied numbers.

LISTING 9-1

```
100  REM       MEANS
110  HOME
120  PRINT "     THIS PROGRAM COMPUTES THE "
130  PRINT : PRINT "ARITHMETIC, GEOMETRIC,AND HARMONIC"
140  PRINT : PRINT "MEANS OF USER-SUPPLIED NUMBERS."
150  VTAB 20
160  PRINT "PRESS ANY KEY TO CONTINUE."
170  GET ANS$
180  HOME
190  PRINT "FOR HOW MANY NUMBERS DO YOU"
200  PRINT : PRINT "WANT THE MEANS";
210  INPUT N
220  PRINT : PRINT
230  DIM A(N)
240  REM  S = RUNNING SUM OF NUMBERS
250  REM  R = RUNNING SUM OF RECIPROCALS
260  REM  P = RUNNING PRODUCT OF NUMBERS
270  REM  INITIALIZE S,R,P:
280 S = 0:R = 0:P = 1
290  PRINT : PRINT "PLEASE ENTER THE NUMBERS."
300  PRINT
310  FOR K = 1 TO N
320  PRINT "A(";K;")= ";
330  INPUT A(K)
340  REM  UPDATE S,R,P:
350 S = S + A(K)
360 R = R + (1 / A(K))
370 P = P * A(K)
380  NEXT
390  HOME
400  PRINT "FOR THE NUMBERS"
410  PRINT
420  FOR K = 1 TO N
430  PRINT A(K),
440  NEXT
450  PRINT : PRINT
460  PRINT : PRINT "THE ARITHMETIC MEAN IS ";S / N
470  PRINT : PRINT "THE GEOMETRIC MEAN IS ";P ^ (1 / N)
480  PRINT : PRINT "THE HARMONIC MEAN IS ";N / R
490  END
```

Lines 100 to 170 print a greeting.

Lines 180 to 230 find out how many numbers are involved and set up an array A(K) to hold the numbers. We store the numbers in an array so that they can be printed out at the end with the three means.

To compute the means, we must compute the sum and product of the numbers and the sum of their reciprocals. We do this by computing running sums and products as the numbers are entered. You will make use of this method of updating sums and products in subsequent programs, so let's analyze the process:

Line 280 initializes the variables S for the running sum of the numbers, R for the running sum of the reciprocals, and P for the running product of the numbers.

Line 330 obtains a number from the user.

Line 350 updates the sum of the numbers by adding the new arrival to S.

Line 360 updates the sum R of reciprocals by adding in the reciprocal of the new number.

Line 370 updates the product P of the numbers.

When the FOR-NEXT loop 310 to 380 is finished, all the numbers have been stored in the array, and S, R, and P hold the desired sums and product.

Lines 400 to 440 then print out the given numbers.

Finally, lines 460 to 480 compute and print the means.

Line 460 gets the arithmetic mean by dividing the sum S by the number of entries N.

Line 470 takes the Nth root of the product P to get the geometric mean.

Line 480 divides the sum R of reciprocals by N to get their average, then reciprocates again to get the harmonic mean.

Exercise 9-9. Try MEANS on various sequences of numbers.

These extended means arise in equalization problems similar to the two-number versions.

Example 9-4. The gymnastics, volleyball, and tennis teams received $1,200, $750, and $900 in last year's budget. If the teams share equally the same total amount of money this year, how much does each team receive? The answer is the *arithmetic* mean(1,200 + 750 + 900)/3.

Example 9-5. The dingbat population has been observed at various times and has shown an increase each time. From the first to the second observation, the population increased by a factor of 1.3. Over the second time interval, it increased by a factor of 1.5. Over the third and fourth intervals, it increased by factors of 1.8 and 2, respectively. If the population had increased by the same factor in each interval for the same total increase, what would that constant factor be?

The answer is the *geometric* mean of the growth factors. If the original population was P, at the end of the first interval it was P*1.3. At the end of the second, it was P*1.3*1.5. At the end of the third, P*1.3*1.5*1.8. At the end of the fourth, P*1.3*1.5*1.8*2. So, P increased by a factor of 1.3*1.5*1.8*2 all together. For equalized growth, we need a growth factor x, such that

$$P*x*x*x*x = P*1.3*1.5*1.8*2$$

Hence, the constant factor is

$$x = \sqrt[4]{1.3*1.5*1.8*2}$$

the geometric mean.

Example 9-6. The Zero manufacturing company has seven gizmo producing machines. Each machine produces 5,000 gizmos per shift, but at different rates, so that they finish at different times. The rates for each machine are listed in Table 9-1.

Table 9-1.

Machine	Rate (Gizmos/Hour)
1	700
2	750
3	825
4	730
5	900
6	860
7	770

You must adjust the machines so that they all run at the same rate and they complete the 35,000 gizmos in the same total amount of machine time. How fast should each machine run?

The answer is the *harmonic* mean of the machine speeds. The first machine runs at 700 gizmos per hour, so it takes 5,000/700 hours to finish. The second takes 5,000/750 hours, and so on. The total machine time is then the sum

$$\begin{aligned} T &= \frac{5{,}000}{700} + \frac{5{,}000}{750} + \frac{5{,}000}{825} + \frac{5{,}000}{730} + \frac{5{,}000}{900} + \frac{5{,}000}{860} + \frac{5{,}000}{770} \\ &= 5{,}000\left(\frac{1}{700} + \frac{1}{750} + \frac{1}{825} + \frac{1}{730} + \frac{1}{900} + \frac{1}{860} + \frac{1}{770}\right) \end{aligned}$$

hours. For the machines to operate at the same speed, they must share this time T equally. So, each must run for 1/7 of this total time, or $T/7$ hours.

$$\frac{T}{7} = 5{,}000\left(\frac{1}{7}\right)\left(\frac{1}{700} + \frac{1}{750} + \frac{1}{825} + \frac{1}{730} + \frac{1}{900} + \frac{1}{860} + \frac{1}{770}\right)$$

Each machine must produce 5,000 gizmos in this time, so each must produce

$$\frac{5{,}000}{\frac{T}{7}}$$

gizmos per hour. When 5,000 is divided by $T/7$, the 5,000s cancel, leaving

$$\frac{1}{\left(\frac{1}{7}\right)\left(\frac{1}{700} + \frac{1}{750} + \frac{1}{825} + \frac{1}{730} + \frac{1}{900} + \frac{1}{860} + \frac{1}{770}\right)}$$

which is the harmonic mean of the given rates.

Exercise 9-10. Use the program MEANS to compute the answer in each of Examples 4 through 6 above.

In addition to solving applied problems, you can use the program MEANS to discover some surprising properties of these means. The following exercises lead you in that direction.

Exercise 9-11. Computer the means for

1. 2, 3, 4
2. 2, 3, 4, 5
3. 2, 4, 6
4. 2, 4, 6, 8
5. 5, 8, 11, 14, 17
6. 5, 8, 11, 14, 17, 20

Make a conjecture. Prove it.

Exercise 9-12. Compute the means for

1. 2, 4, 8
2. 2, 4, 8, 16
3. 3, 12, 48
4. 3, 12, 48, 192

Make a conjecture. Prove it.

Exercise 9-13. Make up this exercise yourself following the pattern of Exercises 9-11 and 9-12.

10

HARMONIC MEAN OF FRACTIONS

In Chapter 9, ''Means,'' we investigated the properties of the arithmetic, geometric, and harmonic means. The program used in that chapter computed the means as decimals, which tend to obscure the behavior of the harmonic mean. In this chapter, you write a program to compute the harmonic mean using only fraction arithmetic. You then use your program for further investigations of the harmonic mean.

The harmonic mean is the reciprocal of the average of the reciprocals of the numbers. Hence, your program must be able to find the reciprocal of a fraction, add fractions (to get the sum for the average), and multiply fractions (to divide the sum by the number of terms in the average).

As did previous programs for working with fractions, the program will carry the numerators and denominators separately. Thus it will be easy to compute a reciprocal: Just switch numerator and denominator.

For the adding and multiplying of fractions, you need various subroutines. These are provided on the class disk as FRACTION SUBROUTINES (see Listing 10-1 at the end of this chapter). This collection of subroutines is also used in other programs that work with fractions. To make use of the subroutines you must know what they do, what variables they use (so you can avoid duplication), and the line numbers where they start. We have worked with most of the subroutines, so you already know what they do.

When you run FRACTION SUBROUTINES, lines 100 to 660 print the information about the subroutines:

```
                FRACTION SUBROUTINES

   GCD:       A, B <> 0 IN, GCD OUT
              USES VARIABLES Q AND R
```

```
REDUCE:     NUM, DEN IN,
            NNUM, NDEN OUT.
            CALLS GCD SUBROUTINE
MULTIPLY:   N1/D1, N2/D2 IN,
            N3/D3 OUT.
            CALLS REDUCE SUBROUTINE.
ORDER:      N1/D1, N2/D2 IN,
            MARK = -1,0,1 OUT.
            USES VARIABLES A, B
ADD:        N1/D1, N2/D2 IN,
            N3/D3 OUT.
            CALLS REDUCE SUBROUTINE.
GETTER:     RETURNS NUM, DEN.
            USES VARIABLES
            VV$, A$, K, CHAR$.
            NUM$, DEN$
          STARTING LOCATION
           GCD: 670
           REDUCE: 790
           ADD: 1160
           MULTIPLY: 920
           ORDER: 1030
           GETTER: 1270
           LAST LINE: 1500
```

Make use of this information for writing your program. For example, the GETTER subroutine uses the variable K, so you should not use K in the part you write.

For a second example, look at the description of the ADD subroutine. It shows that the subroutine expects to be given fractions N1/D1 and N2/D2 and that it returns the sum as N3/D3. Thus, when you want to add two fractions, you must set N1 and D1 equal to the numerator and denominator of one fraction, and N2 and D2 equal to the numerator and denominator of the other.

Finally, the last line information is important. You must use line numbers that are large enough so that you don't destroy the given subroutines.

Exercise 10-1. Write a program that uses the fraction subroutines to compute the harmonic mean of fractions. Follow the outline given below:

1. Find out from the user how many fractions are involved. Record the answer as N, and set up arrays N(J) and D(J) to hold the numerators and denominators.
2. Set up variables NS and DS for the numerator and denominator of the running sum of the reciprocals.
3. For J = 1 to N, use the fraction-reading subroutine to get each fraction. Record the numerator as N(J) and the denominator as D(J).

4. As each fraction N(J)/D(J) is obtained, update the sum of reciprocals by adding (note the reciprocal!) D(J)/N(J) to NS/DS.
5. After all the fractions have been entered, NS/DS is the sum of the reciprocals. Multiply this by 1/N to obtain the average of the reciprocals. Then reciprocate this product to get the harmonic mean.
6. Finally, print the original fractions and their harmonic mean.

Exercise 10-2. When your harmonic mean program is working, try it on the following:

1. $\frac{1}{2}$, $\frac{1}{3}$
2. $\frac{1}{3}$, $\frac{1}{5}$
3. $\frac{1}{7}$, $\frac{1}{10}$
4. $\frac{1}{4}$, $\frac{1}{8}$

Find a simple formula for the harmonic mean of two fractions of the form 1/n, 1/m. Prove that your formula always works.

Exercise 10-3. Compute the harmonic mean of

1. $\frac{3}{5}$, $\frac{3}{17}$
2. $\frac{5}{9}$, $\frac{5}{14}$
3. $\frac{7}{2}$, $\frac{7}{9}$
4. $\frac{9}{4}$, $\frac{9}{11}$

Find a simple formula for computing the harmonic mean of any two fractions of this form. Prove that your formula always works.

It is very difficult to guess this formula for the harmonic mean if the program works in decimals. You would have to enter the fractions as decimals and guess the fraction form of the answer from its decimal representation. So, the hard work of constructing a fraction arithmetic program is paying off.

Exercise 10-4. Use your formula of Exercise 10-3 to devise a method for computing the harmonic mean of any two fractions.

Exercise 10-5. Compute the harmonic mean for each of the following sequences:

1. $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$
2. $\frac{1}{2}$, $\frac{1}{4}$
3. $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, $\frac{1}{5}$, $\frac{1}{6}$

4. 1⁄2, 1⁄3, 1⁄5, 1⁄6
5. 1⁄2, 1⁄3, 1⁄4, 1⁄5
6. 1⁄3, 1⁄4

Make conjectures. Prove them.

Exercise 10-6. Compute the harmonic mean for each of the following:

1. 1⁄2, 1⁄5, 1⁄8
2. 1⁄2, 1⁄5, 1⁄8, 1⁄11
3. 1⁄3, 1⁄10, 1⁄17
4. 1⁄3, 1⁄10, 1⁄17, 1⁄24

Make conjectures. Prove them.

Exercise 10-7. Compute the harmonic mean for each of the following sequences:

1. 1⁄2, 2⁄7, 1⁄5
2. 1⁄2, 2⁄7, 1⁄5, 2⁄13
3. 3⁄4, 1⁄3, 3⁄14
4. 3⁄4, 1⁄3, 3⁄14, 3⁄19

Make conjectures. Prove them.

Exercise 10-8. Based on your experience here with the harmonic mean of sequences of fractions and on your earlier work with arithmetic and geometric means in Exercises 9-11 and 9-12, formulate a definition of a harmonic progression.

LISTING 10-1

```
90  REM     FRACTION SUBROUTINES
100  HOME
110 H = 14: REM    HTAB
120  HTAB 10: PRINT "FRACTIONS SUBROUTINES"
130  PRINT : PRINT "GCD :";
140  HTAB H: PRINT "A, B <> 0 IN, GCD OUT"
150  PRINT
160  HTAB H: PRINT "USES VARIABLES Q AND R."
170  PRINT : PRINT "REDUCE:";
180  HTAB H: PRINT "NUM,DEN IN,"
190  PRINT
200  HTAB H: PRINT "NNUM,NDEN OUT."
210  PRINT
220  HTAB H: PRINT "CALLS GCD SUBROUTINE."
230  PRINT : PRINT "MULTIPLY:";
240  HTAB H: PRINT "N1/D1, N2/D2 IN,"
250  PRINT
```

```
260  HTAB H: PRINT "N3/D3 OUT."
270  PRINT
280  HTAB H: PRINT "CALLS REDUCE SUBROUTINE."
290  VTAB 21: PRINT "PRESS ANY KEY TO CONTINUE."
300  GET ANS$
310  HOME
320  PRINT "ORDER:";
330  HTAB H
340  PRINT "N1/D1, N2/D2 IN,"
350  PRINT
360  HTAB H: PRINT "MARK = -1,0,1 OUT."
370  PRINT
380  HTAB H: PRINT "USES VARIABLES A,B."
390  PRINT
400  PRINT "ADD:";
410  HTAB H: PRINT "N1/D1, N2/D2 IN,"
420  PRINT
430  HTAB H: PRINT "N3/D3 OUT."
440  PRINT
450  HTAB H: PRINT "CALLS REDUCE SUBROUTINE."
460  PRINT
470  PRINT "GETTER:";
480  HTAB H: PRINT "RETURNS NUM, DEN."
490  PRINT
500  HTAB H: PRINT "USES VARIABLES"
510  PRINT
520  HTAB H: PRINT "VV$, A$, K, LT$, RT$,"
530  PRINT
540  HTAB H: PRINT "NUM$, DEN$"
550  VTAB 21: PRINT "PRESS ANY KEY TO CONTINUE."
560  GET ANS$
570  HOME
580  HTAB 10: PRINT "STARTING LOCATIONS"
590  VTAB 3: PRINT "GCD: 670"
600  VTAB 5: PRINT "REDUCE: 790"
610  VTAB 7: PRINT "ADD: 1160"
620  VTAB 9: PRINT "MULTIPLY: 920"
630  VTAB 11: PRINT "ORDER: 1030"
640  VTAB 13: PRINT "GETTER: 1270"
650  VTAB 15: PRINT "LAST LINE: 1500"
660  END
670  REM  GCD SUBROUTINE:A,B IN,GCD OUT
680  REM  SUBROUTINE ASSUMES B<>0
690 Q =  INT (A / B): REM  DIVIDE A BY B
700 R = A - Q * B: REM  REMAINDER
710  IF R = 0 THEN 750: REM   ALGORITHM FINISHED,GCD IS B
720  REM  IF R <> 0 MUST DO ANOTHER DIVISION
730  REM  NOW SET UP FOR NEXT DIVISION
740 A = B:B = R: GOTO 690
750 GCD = B
760  RETURN
770  REM  SUBROUTINE FINISHED
780  REM
790  REM   SUBROUTINE TO REDUCE FRACTION
800  REM  USING GCD SUBROUTINE
810  REM    NUM, DEN IN, NNUM,NDEN OUT
820  REM  IF NUM = 0, SKIP GCD SUBRTN
830  IF NUM = 0 THEN NNUM = 0:NDEN = 1: RETURN
840 A = NUM:B = DEN
850  GOSUB 690: REM  GCD SUBROUTINE
860  REM  SUBROUTINE RETURNS GCD
870  REM   NOW DIVIDE OUT BY GCD:
880 NNUM = NUM / GCD:NDEN = DEN / GCD
890  REM  REDUCED FORM IS NNUM/NDEN
900  RETURN
910  REM
```

```
920  REM  SUBROUTINE TO MULTIPLY FRACS
930  REM  N1/D1 AND N2/D2 IN, N3/D3 OUT
940  REM
950  REM  COMPUTE UNREDUCED PRODUCT NUM/DEN
960 NUM = N1 * N2
970 DEN = D1 * D2
980  GOSUB 790: REM   REDUCE TO NNUM/NDEN
990  REM  SET UP FOR RETURN
1000 N3 = NNUM:D3 = NDEN
1010  RETURN
1030  REM  SUBROUTINE TO ORDER FRACTIONS
1040  REM  N1/D1 AND N2/D2 IN
1050  REM  RETURNS:
1060  REM              MARK=-1 IF N1/D1<N2/D2
1070  REM              MARK = 0 IF N1/D1=N2/D2
1080  REM            MARK=1 IF N1/D1>N2/D2
1090  REM  CROSS MULTIPLY:
1100 A = N1 * D2:B = N2 * D1
1110  IF A < B THEN MARK =  - 1
1120  IF A = B THEN MARK = 0
1130  IF A > B THEN MARK = 1
1140  RETURN
1150  REM
1160  REM  SUBROUTINE TO ADD FRACTIONS
1170  REM  N1/D1 AND N2/D2 IN,
1180  REM  SUM N3/D3 OUT
1190  REM  FORM UNREDUCED SUM:
1200 NUM = N1 * D2 + N2 * D1
1210 DEN = D1 * D2
1220  IF NUM = 0 THEN N3 = 0:D3 = 1: GOTO 1250: REM  SKIP REDU
CE
1230  GOSUB 790: REM  REDUCE TO NNUM/NDEN
1240 N3 = NNUM:D3 = NDEN
1250  RETURN
1260  REM
1270  REM  SUBRTN TO READ FRAC FROM KYBD
1280  REM  SUBRTN EXPECTS STRING NUM/DEN
1290  REM         AND EXTRACTS NUM AND DEN
1300  REM  BEFORE ENTERING SBRTN, SET
1310  REM   VV$= SOME CONNECTING WORD
1320  PRINT : PRINT "PLEASE TYPE ";VV$;"FRACTION"
1330  INPUT A$
1340  REM  SEARCH FOR "/" IN A$:
1350  FOR K = 1 TO  LEN (A$)
1360 LT$ =  LEFT$ (A$,K): REM  LEFT K CHARS OF A$
1370 RT$ =  RIGHT$ (LT$,1): REM  RIGHTMOST CHAR OF LT$
1380  IF RT$ = "/" THEN 1450: REM  FOUND "/"
1390  NEXT : REM  KEEP LOOKING FOR "/"
1400  REM  HERE, A$ HAS NO "/";ASSUME A$ IS INTEGER
1410 NUM =  VAL (A$): REM  NUMERICAL VAL OF A$
1420 DEN = 1
1430  RETURN
1440  REM  HERE, HAVE FOUND "/" AS K-TH CHAR OF A$
1450 NUM$ =  LEFT$ (A$,K - 1)
1460 DEN$ =  RIGHT$ (A$, LEN (A$) - K)
1470 NUM =  VAL (NUM$)
1480 DEN =  VAL (DEN$)
1490  IF DEN = 0 THEN  PRINT : PRINT "DENOMINATOR NOT ALLOWED
TO BE ZERO.": GOTO 1320
1500  RETURN
```

11

SQUARE ROOTS

In this chapter, we investigate how the computer handles square roots, and we start to develop a method of approximating square roots by fractions. The first thing to realize is that the computer can*not* compute square roots. (At least, not and get the right answer.) To be convinced of this, consider the program SQUARE ROOT DEMO (see Listing 11-1).

LISTING 11-1

```
100  REM      SQUARE ROOT DEMO
110  HOME
120  PRINT "FOR WHAT NUMBER DO YOU WANT"
130  PRINT
140  PRINT "THE SQUARE ROOT";
150  INPUT N
160 Z =  SQR (N)
170  PRINT : PRINT "MY VALUE FOR THE ROOT IS ";Z
180  PRINT
190  IF Z * Z = N THEN  PRINT "AND ITS SQUARE IS ";N: GOTO 230

200  REM  HERE, SQUARE OF Z IS NOT N
210  IF Z * Z > N THEN  PRINT "BUT ITS SQUARE IS LARGER THAN
";N
220  IF Z * Z < N THEN  PRINT "BUT ITS SQUARE IS LESS THAN ";N

230  PRINT : PRINT "DO YOU WANT TO DO ANOTHER ONE (Y OR N)";
240  INPUT ANS$
250  IF ANS$ = "Y" THEN 110
260  END
```

Line 160 applies the computer's square root function to the user-supplied number N and stores the result in the variable Z.

Then line 170 prints out the value of Z.

If Z were really the square root of N, then Z*Z would equal N. Lines 190 to 220 compare Z*Z to N and report the results.

Exercise 11-1. Run SQUARE ROOT DEMO for various numbers to see what happens.

You see by Exercise 11-1 that the computer does not always get the exact value for a square root and, moreover, tries to cover up by printing a rounded-off version of its answer. (Perhaps this program should be called LIE DETECTOR!)

To make the machine's lies even more evident, apply the square root function several times. If you start with the number 5, for example, and apply the square root operation three times, you will get some number Z. If you then square Z three times, you should have 5 back again. You can try this out with the program SQUARE ROOTS REPEATEDLY (see Listing 11-2).

LISTING 11-2

```
100  REM   SQUARE ROOTS REPEATEDLY
110  HOME
120  PRINT "      THIS PROGRAM REPEATEDLY TAKES THE"
130  PRINT : PRINT "SQUARE ROOT OF A NUMBER."
140  PRINT : PRINT "      THEN IT REPEATEDLY SQUARES "
150  PRINT : PRINT "THE RESULT."
160  PRINT : PRINT "       YOU WILL SUPPLY THE STARTING"
170  PRINT : PRINT "NUMBER.
180  PRINT : PRINT "       YOU WILL ALSO SPECIFY THE "
190  PRINT : PRINT "NUMBER OF SQUARE ROOTS TO BE DONE."
200  VTAB 23
210  PRINT "PRESS ANY KEY TO CONTINUE."
220  GET ANS$
230  HOME
240  PRINT "STARTING NUMBER = ";
250  INPUT NUM
260  PRINT : PRINT "TAKE THE SQUARE ROOT HOW MANY TIMES";: INPUT
TIMES
270 Z = NUM
280  HOME
290  PRINT "STARTING WITH THE NUMBER ";NUM
300  PRINT : PRINT "AND TAKING THE SQUARE ROOT K TIME GIVES"
310  PRINT : PRINT " K             SQUARE ROOTS"
320  PRINT
330  FOR K = 1 TO TIMES
340 Z =  SQR (Z)
350  PRINT K,Z
360  NEXT K
370  PRINT : PRINT
380  VTAB 23
390  PRINT "PRESS ANY KEY TO SEE THE SQUARES."
400  GET ANS$
410  HOME
420  PRINT "STARTING WITH THE LAST ROOT ";Z
430  PRINT : PRINT "AND SQUARING K TIMES GIVES"
440  PRINT : PRINT "K                SQUARES:"
450  PRINT
460  FOR K = 1 TO TIMES
470 Z = Z * Z
480  PRINT K,Z
490  NEXT
500  PRINT : PRINT "THE STARTING NUMBER WAS ";NUM
510  PRINT : PRINT
520  VTAB 23: PRINT "DO ANOTHER (Y OR N)";
530  INPUT ANS$
540  IF ANS$ <   > "N" THEN 230
```

Lines 240 to 260 obtain the starting number NUM and the desired number TIMES of repetitions of the square root function.

Lines 330 to 360 compute and print successive square roots.

Lines 460 to 490 then compute and print the corresponding squares.

Exercise 11-2. Run SQUARE ROOTS REPEATEDLY for various numbers and levels of repetition.

Exercise 11-3. For the starting number 5, how many square root repetitions do you need for there to be a visible difference between the program's final number and 5?

Exercise 11-4. Starting with 5, apply the square root operation for more and more times. What happens? Try this for other starting numbers and compare the results.

The difficulty in getting exact values for square roots comes from the fact that the computer can carry only a finite number of decimal places. Hence, it must round off during its computations.

In the exercises above you had some experience with the computer's square root function and saw that the values it gives are not exact. That may seem bad, but what is even worse is that we have no way of knowing how close they are to the true values, because we don't know what the computer does in its square root subroutine.

We will now investigate an alternate (better?) method of approximating square roots that uses bracketing fractions. We say that fractions a/b and c/d *bracket* a number z if

$$\frac{a}{b} \le z \le \frac{c}{d}$$

By working in fractions, we avoid the computer's round-off error, and we also get some of the flavor of predecimal mathematics. (Decimals were not widely used until the sixteenth century C.E.) By using bracketing fractions, we always know an interval that contains the exact value of the square root.

To approximate, say, the square root of ½, we want to find fractions a/b and c/d, such that

$$\left(\frac{a}{b}\right)^2 \le \frac{1}{2} \le \left(\frac{c}{d}\right)^2$$

Then we know that

$$\frac{a}{b} \le \sqrt{\frac{1}{2}} \le \frac{c}{d}$$

so that a/b and c/d bracket the square root of ½.

Since fractions with large denominators were unwieldy for precomputer mathematicians, they would have insisted that the denominators be "small." So a typical approximation problem would have been as follows: Find the two fractions a/b and c/d, with denominators b and d no greater than 75, which are as close as possible on either side of $\sqrt{1/2}$. A naive way to do this is to note that 0 and 1 bracket the square root:

$$0^2 = 0 < \frac{1}{2}$$

so 0 is smaller than the square root of ½. Since

$$1^2 > \frac{1}{2}$$

1 is larger than the square root.

Thus, write down all the fractions between 0 and 1 with denominators up to 75, put them in order, and search for the two whose squares bracket ½. To do this, we need a subroutine to compare two fractions. We also need a subroutine that puts an array of numbers in numerical order (this is called *sorting* the array).

The first of these requirements is the easier to fulfill. The subroutine ORDER (see Listing 11-3) compares two fractions and returns a number MARK whose value depends on the order relationship of the two fractions. Note that the subroutine does not resort to decimals to make the comparison.

LISTING 11-3

```
10   REM   ORDER
1310   REM   SUBROUTINE TO ORDER FRACTIONS
1320   REM   N1/D1 AND N2/D2 IN
1330   REM   RETURNS:
1340   REM                 MARK=0 IF EQUAL
1350   REM                 MARK=-1 IF N1/D1<N2/D2
1360   REM                 MARK=1 IF N1/D1>N2/D2
1370   REM   CROSS MULTIPLY:
1380 A = N1 * D2:B = N2 * D1
1390   IF A = B THEN MARK = 0
1400   IF A < B THEN MARK =  - 1
1410   IF A > B THEN MARK = 1
1420   RETURN
1430   REM
```

Exercise 11-5. Add some lines to ORDER to make a program that compares two user-supplied fractions. Use your program to compare

1. ½, ¾
2. ⅞, $^3/_{17}$
3. $^5/_{16}$, $^4/_{13}$
4. $^{123}/_{465}$, $^{1107}/_{4185}$

Now for the more difficult part—sorting an array of numbers. The program SORTER (see Listing 11-4) sorts an array A(K), $1 \leq K \leq$ BOUND, of user-supplied numbers. It scans the array from A(1) to A(BOUND). If it finds out-of-order neighbors, it switches them. It repeats the scanning until no further switches are necessary. (This is called a *bubble sort*.)

LISTING 11-4

```
100  REM      SORTER
110  HOME
120  PRINT "     THIS PROGRAM SORTS USER-SUPPLIED"
130  PRINT : PRINT "NUMBERS USING A BUBBLE SORT TECHNIQUE."
140  VTAB (23)
150  PRINT "PRESS ANY KEY TO CONTINUE."
160  GET ANS$
170  HOME
180  PRINT "HOW MANY NUMBERS DO YOU WANT TO SORT";
190  INPUT BOUND
200  REM  SET UP ARRAY TO BE SORTED
210  DIM A(BOUND)
220  PRINT : PRINT "TYPE IN THE NUMBERS, ONE AT A TIME"
230  PRINT
240  FOR K = 1 TO BOUND
250  PRINT "A(";K;") = ";: INPUT A(K)
260  NEXT
270  REM :USE FLAG TO RECORD PRESENCE OF INTERCHANGE
280 FLAG = 0: REM  INITIALIZE FLAG
290  FOR J = 1 TO BOUND - 1
300  IF A(J) <  = A(J + 1) THEN 360: REM   SKIP INTERCHANGE

310  REM  INTERCHANGE A(J) AND A(J+1)
320 TEMP = A(J): REM  HOLD VALUE OF A(J)
330 A(J) = A(J + 1)
340 A(J + 1) = TEMP
350 FLAG = 1: REM  RECORD INTERCHANGE
360  NEXT
370  REM  IF FLAG = 1, AN INTERCHANGE WAS MADE IN LAST PASS.
OTHERWISE, FLAG = 0
380  IF FLAG = 1 THEN 280: REM  MORE INTERCHANGES MAY BE NEEDE
D, DO ANOTHER PASS
390  REM  HERE, FLAG = 0;NO INTERCHANGES WERE NEEDED;SORT COMP
LETE
400  REM  PRINT SORTED ARRAY
410  FOR K = 1 TO BOUND
420  PRINT A(K);" ";
430  NEXT
```

Lines 240 to 260 obtain the numbers to be sorted and put them in an array.

Line 280 initializes the variable FLAG to 0. It is set equal to 1 if a switch is made.

The FOR-NEXT loop in lines 290 to 360 looks for out-of-order neighbors.

Lines 320 to 340 switch out-of-order neighbors, and line 350 sets the flag to indicate that a switch was made.

The process stops when a pass is made through this FOR-NEXT loop with no interchanges being made. See lines 370 and 380.

Exercise 11-6. Run SORTER for various lists of numbers to see that it works.

Exercise 11-7. Add some lines to SORTER so that it prints the array each time an interchange is made, and prints a message, such as ''PASS COMPLETED'' after each completion of the FOR-NEXT loop in lines 290 to 360. Run this version for 9, 5, 6, 7, 8. Examine the printout to see the interchanges and how the higher numbers bubble up to the top. Repeat for 9, 8, 7, 6, 5.

Exercise 11-8. Modify SORTER so that it puts the numbers in reverse order that is, from largest to smallest.

Now we're about ready to sort fractions. We first need an array of all the fractions between 0 and 1 with denominators up to a user-supplied largest denominator, DMAX. We then apply the bubble sort technique with the fraction-ordering subroutine to sort the array.

In order to be sure we have all the fractions, we must consider each fraction N/D, for D = 1 to DMAX. For each such D, N must run from 0 to D. To avoid repetition in the array of fractions, we put in only those fractions N/D that are in lowest terms. That is, we put N/D in the array if and only if GCD (N, D) = 1.

The program FRACTION SORTER (see Listing 11-5) constructs an ordered array of fractions as we have just described.

LISTING 11-5

```
100  REM    FRACTION SORTER
110  HOME
120  PRINT "     THIS PROGRAM SORTS AND PRINTS"
130  PRINT
140  PRINT "ALL FRACTIONS BETWEEN 0 AND 1 WITH"
150  PRINT
160  PRINT "DENOMINATORS UP TO A USER-SUPPLIED"
170  PRINT
180  PRINT "BOUND."
190  GOTO 470
200  REM
210  REM  GCD SUBROUTINE:A,B IN,GCD OUT
220  REM  SUBROUTINE ASSUMES B<>0
230 Q =  INT (A / B): REM  DIVIDE A BY B
240 R = A - Q * B: REM  REMAINDER
250  IF R = 0 THEN 290: REM   ALGORITHM FINISHED,GCD IS B
260  REM  IF R <> 0 MUST DO ANOTHER DIVISION
270  REM  NOW SET UP FOR NEXT DIVISION
280 A = B:B = R: GOTO 230
290 GCD = B
300  RETURN
310  REM  SUBROUTINE FINISHED
320  REM
330  REM
340  REM  SUBROUTINE TO ORDER FRACTIONS
350  REM  N1/D1 AND N2/D2 IN
360  REM  RETURNS:
370  REM            MARK=0 IF EQUAL
380  REM            MARK=-1 IF N1/D1<N2/D2
390  REM            MARK=1 IF N1/D1>N2/D2
400  REM  CROSS MULTIPLY:
410 A = N1 * D2:B = N2 * D1
420  IF A = B THEN MARK = 0
430  IF A < B THEN MARK =  - 1
440  IF A > B THEN MARK = 1
450  RETURN
460  REM
470  REM  PROG TO SORT AND PRINT FRACTIONS
480  VTAB 15: PRINT "WHAT DO YOU WANT"
490  PRINT : PRINT "THE LARGEST DENOMINATOR TO BE";
500  INPUT DMAX
510  HOME
```

```
520  REM  COMPUTE ADEQUATE SIZE FOR ARRAY
530 SIZE = DMAX * (DMAX + 1)
540  DIM N(SIZE),D(SIZE)
550  REM  SET UP UNORDERED ARRAY
560  PRINT "LOADING ARRAY OF FRACS, 0 <= N/D <= 1,"
570  PRINT : PRINT "DENOMINATORS UP TO ";DMAX
580  PRINT
590 J = 1: REM   SET INDEX FOR ARRAY
600  FOR D = 1 TO DMAX
610  FOR N = 0 TO D
620  REM  SET UP FOR GCD SUBROUTINE
630 A = N:B = D
640  GOSUB 200: REM  FIND GCD
650  IF GCD > 1 THEN 700: REM   NOT IN LOWEST TERMS, ALREADY
IN ARRAY
660  REM   HERE, N/D IN LOWEST TERMS, SO PUT IN ARRAY
670 N(J) = N:D(J) = D
680  PRINT "PUTTING  ";N(J);"/";D(J);"  IN ARRAY"
690 J = J + 1: REM  ADVANCE INDEX
700  NEXT N
710  NEXT D
720  REM
730  REM
740  REM   RECORD J-1, LENGTH OF ARRAY
750 LAST = J - 1
760  REM :ARRAY OF FRACS COMPLETE
770  REM :NOW SORT THE ARRAY
780  PRINT
790  PRINT "PUTTING THE ARRAY IN ORDER"
800  PRINT
810  REM
820 FLAG = 0: REM  INITIALIZE FLAG
830  FOR J = 1 TO LAST - 1
840  REM  SET UP FOR ORDER SUBROUTINE
850 N1 = N(J):D1 = D(J):N2 = N(J + 1):D2 = D(J + 1)
860  GOSUB 330: REM  ORDER, MARK RETURNS
870  IF MARK <  = 0 THEN 930: REM   NO SWITCH
880  REM   HERE, MARK > 0, SO SWITCH
890 T1 = N(J):T2 = D(J)
900 N(J) = N(J + 1):D(J) = D(J + 1)
910 N(J + 1) = T1:D(J + 1) = T2
920 FLAG = 1: REM  RECORD THE SWITCH
930  NEXT : REM  GO TO NEXT FRAC PAIR
940  REM  IF FLAG = 1 HERE, A SWITCH WAS MADE, SO ANOTHER PASS
 MAY BE NEEDED
950  IF FLAG = 1 THEN 810
960  REM  HERE, SORT IS COMPLETE
970  REM  NOW PRINT SORTED ARRAY
980  HOME
990  PRINT "HERE IS SORTED ARRAY:"
1000  PRINT
1010  FOR K = 1 TO LAST
1020  PRINT N(K);"/";D(K);"    ";
1030  NEXT
```

Lines 210 to 300 and 340 to 450 contain the GCD and ORDER subroutines.

Lines 480 to 500 obtain the largest denominator DMAX.

Lines 530 to 540 set up arrays to hold the numerators and denominators. These arrays must be large enough to hold all the numerators and denominators. So what must their dimension be? Since D runs from 1 to DMAX, there are DMAX denominators. For each D, N runs from 0 to D, which gives D + 1 values for N. Since D ≤ DMAX, there are fewer than DMAX + 1 values of N for each D. Altogether, then, there are fewer than

DMAX*(DMAX + 1) pairs (N, D). This is the value given to the dimension SIZE in line 530.

Lines 600 to 710 put the numerators and denominators in the arrays, using the GCD subroutine to screen out unreduced fractions.

Lines 820 to 960 apply the bubble sort to the fractions.

Finally, lines 990 to 1030 print out the fractions in order.

Exercise 11-9. Run FRACTION SORTER for DMAX = 4, 5, 6, and 7 to see that it works correctly. Try it on some larger values of DMAX to see how slow it is.

Exercise 11-10. The value given to SIZE in line 530 is adequate for the arrays of fractions we have to work with. In fact, it is rather generous. To see this, note that line 750 records the actual number of fractions as LAST. Add lines to FRACTION SORTER to print the values of SIZE and LAST. Then run the program for various values of DMAX to see by how much SIZE is too large. You may wish to skip the ordering part of the program to save time.

Who cares if the array set up to hold the fractions is too large? Well, if you wanted to work with really large DMAX, you might run out of memory in the computer.

Exercise 11-11. Try to find a smaller value for SIZE that will still give arrays large enough to hold all the fractions.

We want to use this fraction-sorting program to approximate square roots. So far, we have worked only with fractions between 0 and 1. For wider applications, you would want to be able to order fractions between any two integers, as in the following exercise.

Exercise 11-12. Modify FRACTION SORTER so that it lists, in order, the fractions between two user-supplied integers LOW and HIGH. (Not just between 0 and 1.)

The program FRACTION ROOT SEARCH uses FRACTION SORTER to approximate square roots. It first constructs an ordered array of all fractions between two integers LOW and HIGH with denominators up to a user-supplied bound. Then it searches the array for the two fractions whose squares bracket the given target fraction. The bound on the denominators is DMAX. The target fraction, whose square root is sought, is NTARG/DTARG. Listing 11-6 contains the portion of the program after the ordered array of fractions is constructed.

LISTING 11-6

```
810  PRINT "      FOR WHAT FRACTION DO YOU WANT TO"
820  PRINT : PRINT "APPROXIMATE THE SQUARE ROOT?"
830  PRINT : PRINT "NUMERATOR = ";: INPUT NTARG
840  PRINT : PRINT "DENOMINATOR = ";: INPUT DTARG
850  HOME
860  PRINT "AMONG ALL FRACTIONS BETWEEN ";LOW;" AND ";HIGH
870  PRINT : PRINT "WITH DENOMINATORS UP TO ";DMAX
880  PRINT : PRINT "THE CLOSEST BRACKETING IS GIVEN BY"
890  PRINT
900  REM  SEARCH ARRAY FOR ENTRIES WHOSE
```

```
910  REM  SQUARES BRACKET NTARG/DTARG
920  REM  SET UP TARGET FOR ORDER SUBRTN
930 N2 = NTARG:D2 = DTARG
940  REM  BEGIN SEARCH
950  FOR J = 1 TO LAST
960  REM  SQUARE N(J)/D(J) AND SET FOR ORDER SUBRTN
970 N1 = N(J) * N(J):D1 = D(J) * D(J)
980  GOSUB 1320: REM  ORDER, MARK RETURNS
990  IF MARK < 0 THEN 1020: REM   NEXT J
1000  IF MARK = 0 THEN 1050: REM   HIT TARGET!
1010  IF MARK > 0 THEN 1070: REM   HAVE FOUND FIRST ENTRY LARG
ER THAN TARGET
1020  NEXT J
1030  REM  HERE, EXHAUSTED ARRAY WITHOUT FINDING ONE LARGER
THAN TARGET
1040  PRINT "SQUARE OF ";N(LAST);"/";D(LAST);" < ";NTARG;"/";D
TARG: GOTO 1130
1050  REM  HERE, HAVE HIT TARGET
1060  PRINT "SQUARE OF ";N(J);"/";D(J);" = ";NTARG;"/";DTARG:
GOTO 1130
1070  REM  HERE, HAVE FIRST ENTRY, N(J)/D(J) LARGER THAN TARGE
T. IF J = 1, ALL ENTRIES ARE LARGER THAN TARGET
1080  PRINT
1090  IF J = 1 THEN 1110
1100  PRINT "SQUARE OF ";N(J - 1);"/";D(J - 1);" < ";
1110  PRINT NTARG;"/";DTARG;
1120  PRINT " < ";"SQUARE OF ";N(J);"/";D(J)
1130  VTAB 23: PRINT "DO YOU WANT TO DO ANOTHER (Y OR N )";
1140  INPUT ANS$
1150  IF ANS$ <  > "Y" THEN  PRINT : PRINT "SO LONG!": END
1160  HOME
1170  GOTO 810
```

Exercise 11-13. Run FRACTION ROOT SEARCH for denominators up to 3 between LOW = 1 and HIGH = 2. Find the bracketing for the square root of 2. The result is

$$\text{square of } \frac{4}{3} \leq \frac{2}{1} \leq \text{square of } \frac{3}{2}$$

This means that

$$\frac{4}{3} \leq \text{square root of } 2 \leq \frac{3}{2}$$

The decimal equivalents of these bracketing fractions are 1.3333 . . . and 1.5, so we know that the square root of 2 is between these numbers.

Exercise 11-14. Use FRACTION ROOT SEARCH to find the fractions with denominators up to DMAX that most closely bracket the square root of N for the following:

1. N = 2, DMAX = 5
2. N = 3, DMAX = 5
3. N = 5/2, DMAX = 5
4. N = 2, DMAX = 10

5. N = 3, DMAX = 10
6. N = 5/2, DMAX = 10

In each case, compare the decimal representations of the bracketing fractions to the computer's decimal value for the square root.

Exercise 11-15. Use this program to find a fraction bracketing for the square root of 10. Find, by hand, the proper integers LOW and HIGH to use. Use denominators up to 5. Repeat for the square root of 20.

The program FRACTION ROOT SEARCH takes quite a long time to construct the array of fractions. You can avoid some of this waiting time by patiently constructing an array with denominators up to 50, for instance, and saving the array on a disk. Then you can read the array in from the disk each time you want to do a square root approximation. However, in the next chapter we develop a method for approximating square roots by fractions, which is even more efficient than this.

Finally, we investigate some of the ideas used in the next chapter. The *Farey sequence* F_n is the sequence of all fractions with denominators up to n. When we look for a bracketing by fractions with denominators up to n, we are working with F_n.

Exercise 11-16. For the first four Farey sequences, here are the fractions that lie between 0 and 1:

F_1	0/1						1/1
F_2	0/1			1/2			1/1
F_3	0/1		1/3	1/2	2/3		1/1
F_4	0/1	1/4	1/3	1/2	2/3	3/4	1/1
F_5							
F_6							
F_7							
F_8							
F_9							

Use FRACTION SORTER to complete the table.

Exercise 11-17. For each n, compute the difference between consecutive entries in the Farey sequence F_n. Look for a formula.

Exercise 11-18. Given two neighboring fractions in a Farey sequence F_n, find out how to compute the first fraction that will come in to separate them in a later sequence.

Exercise 11-19. Use the result of Exercise 11-18 to devise a more efficient root-search method.

12

FAREY SEQUENCES AND ROOTS

In the Chapter 11, ''Square Roots,'' we used a naive but slow method of approximating square roots by fractions. Here we develop a much more efficient method. This method relies on Farey sequences, which you met in the preceding chapter. Figure 12-1 contains portions of the first few Farey sequences.

$$F_1 \quad \frac{0}{1} \quad \frac{1}{1}$$

$$F_2 \quad \frac{0}{1} \quad \frac{1}{2} \quad \frac{1}{1}$$

$$F_3 \quad \frac{0}{1} \quad \frac{1}{3} \quad \frac{1}{2} \quad \frac{2}{3} \quad \frac{1}{1}$$

$$F_4 \quad \frac{0}{1} \quad \frac{1}{4} \quad \frac{1}{3} \quad \frac{1}{2} \quad \frac{2}{3} \quad \frac{3}{4} \quad \frac{1}{1}$$

$$F_5 \quad \frac{0}{1} \quad \frac{1}{5} \quad \frac{1}{4} \quad \frac{1}{3} \quad \frac{2}{5} \quad \frac{1}{2} \quad \frac{3}{5} \quad \frac{2}{3} \quad \frac{3}{4} \quad \frac{4}{5} \quad \frac{1}{1}$$

$$F_6 \quad \frac{0}{1} \quad \frac{1}{6} \quad \frac{1}{5} \quad \frac{1}{4} \quad \frac{1}{3} \quad \frac{2}{5} \quad \frac{1}{2} \quad \frac{3}{5} \quad \frac{2}{3} \quad \frac{3}{4} \quad \frac{4}{5} \quad \frac{5}{6} \quad \frac{1}{1}$$

$$F_7 \quad \frac{0}{1} \quad \frac{1}{7} \quad \frac{1}{6} \quad \frac{1}{5} \quad \frac{1}{4} \quad \frac{2}{7} \quad \frac{1}{3} \quad \frac{2}{5} \quad \frac{3}{7} \quad \frac{1}{2} \quad \frac{4}{7} \quad \frac{3}{5} \quad \frac{2}{3} \quad \frac{5}{7} \quad \frac{3}{4} \quad \frac{4}{5} \quad \frac{5}{6} \quad \frac{6}{7} \quad \frac{1}{1}$$

$$F_8 \quad \frac{0}{1} \quad \frac{1}{8} \quad \frac{1}{7} \quad \frac{1}{6} \quad \frac{1}{5} \quad \frac{1}{4} \quad \frac{2}{7} \quad \frac{1}{3} \quad \frac{3}{8} \quad \frac{2}{5} \quad \frac{3}{7} \quad \frac{1}{2} \quad \frac{4}{7} \quad \frac{3}{5} \quad \frac{5}{8} \quad \frac{2}{3} \quad \frac{5}{7} \quad \frac{3}{4} \quad \frac{4}{5} \quad \frac{5}{6} \quad \frac{6}{7} \quad \frac{7}{8} \quad \frac{1}{1}$$

Figure 12-1 Farey sequences

Note the following properties of the sequences: (A) If a/b and c/d are neighbors in a Farey sequence, the first fraction that comes in to separate them in a later sequence is their *mediant* (a + b) / (c + d), and (B) If a/b and c/d are neighboring fractions in a Farey sequence, the difference between them is 1/bd.

Exercise 12-1. Check property A for the sequences in the table.

A warning about computing mediants is in order: If an equivalent unreduced fraction is substituted for a/b or c/d, the formula in property A may give a different value. (Try it!) Hence, you must put fractions in lowest terms before computing their mediant. The fractions in the Farey sequences are in lowest terms anyway, so this causes us no problem here.

Exercise 12-2. Use your fraction arithmetic program from Exercise 8-7 to check property B for the sequence F_8.

Property A gives the basis for a root-search technique. Suppose we're approximating the square root of a target fraction NTARG/DTARG, and we know that the fractions NL/DL and NH/DH bracket this root in a certain Farey sequence F_n:

$$\frac{\mathrm{NL}}{\mathrm{DL}} \leq \frac{\mathrm{NTARG}}{\mathrm{DTARG}} \leq \frac{\mathrm{NH}}{\mathrm{DH}}$$

Then NL/DL is the low estimate (L is for "low"), and NH/DH is the high estimate (H is for "high"). Moreover, since these fractions are neighbors in a Farey sequence, they are the closest fractions to NTARG/DTARG among all fractions with denominators up to n.

As larger denominators are allowed, NL/DL and NH/DH remain the bracketing fractions until some fraction comes in to separate them in a later Farey sequence. By property A, the first fraction to do so is their mediant

$$\frac{\mathrm{NM}}{\mathrm{DM}} = \frac{\mathrm{NL} + \mathrm{NH}}{\mathrm{DL} + \mathrm{DH}}$$

When the mediant does come in, it will be either the new high estimate or the new low estimate. If NM/DM ≤ NTARG/DTARG, take NM/DM as the new low estimate and let NL = NM and DL = DM. If NM/DM ≥ NTARG/DTARG, take NM/DM as the new high estimate and let NH = NM and DH = DM. Now we're back where we started. We have the root bracketed by two neighbors in a Farey sequence. Hence we can repeat the procedure to get the next bracketing.

Once we have the root bracketed by fractions NL/DL and NH/DH from a Farey sequence, we can take either one as an estimate of the root. How large can the error be in using one of these estimates? We know the root is in between them, and property B says that their difference is 1/(DL*DH), so the error is at most 1/(DL*DH).

As an example, let's approximate the square root of $\frac{2}{7}$. Start with bracketing integers:

$$0^2 = 0 < \frac{2}{7}$$

and

$$1^2 = 1 > \frac{2}{7}$$

so the *first bracketing* is

$$\frac{0}{1} < \sqrt{\frac{2}{7}} < \frac{1}{1}$$

The first mediant step is as follows: The mediant of 0/1 and 1/1 is $\frac{1}{2}$. The square of $\frac{1}{2}$ is $\frac{1}{4}$, and $\frac{1}{4} < \frac{2}{7}$, since 1*7 < 2*4. Hence, $\frac{1}{2}$ is less than $\sqrt{2/7}$, so make $\frac{1}{2}$ the new low estimate.

The *second bracketing* is

$$\frac{1}{2} < \sqrt{\frac{2}{7}} < \frac{1}{1}$$

Thus $\frac{1}{2}$ and $\frac{1}{1}$ are the closest fractions to $\sqrt{2/7}$ with the denominators up to 2.

The second mediant step is as follows: The mediant of $\frac{1}{2}$ and $\frac{1}{1}$ is $\frac{2}{3}$. The square of $\frac{2}{3} = \frac{4}{9}$ and $\frac{4}{9} > \frac{2}{7}$, since 4*7 > 9*2. Hence, $\frac{2}{3} > \sqrt{\frac{2}{7}}$, so make $\frac{2}{3}$ the new high estimate.

The *third bracketing* is

$$\frac{1}{2} < \sqrt{\frac{2}{7}} < \frac{2}{3}$$

Now we know that $\frac{1}{2}$ and $\frac{2}{3}$ are the closest fractions to $\sqrt{\frac{2}{7}}$ with denominators no larger than 3.

Another mediant step is as follows: The mediant of $\frac{1}{2}$ and $\frac{2}{3}$ is $\frac{3}{5}$. The square of $\frac{3}{5}$ is $\frac{9}{25}$ and $\frac{9}{25} > \frac{2}{7}$, so $\frac{3}{5} > \sqrt{\frac{2}{7}}$. Make $\frac{3}{5}$ the new high estimate.

The *fourth bracketing* is

$$\frac{1}{2} < \sqrt{\frac{2}{7}} < \frac{3}{5}$$

Now $\frac{1}{2}$ and $\frac{3}{5}$ are the closest fractions with denominators up to 5. Moreover, if we take either $\frac{1}{2}$ or $\frac{3}{5}$ as an estimate for $\sqrt{\frac{2}{7}}$, the error is less than 1/(2*5) = $\frac{1}{10}$.

Now it's your turn: Continue for two more steps. You may want to use your fraction arithmetic program from Exercise 8-7 for the squaring and comparing. At the end, compute the error estimate given by property B. Also convert the bracketing fractions to decimals and compare them to the computer's SQR($\frac{2}{7}$).

Exercise 12-3. Approximate $\sqrt{\frac{17}{7}}$. First find bracketing integers

$$1^2 = 1 > \frac{17}{7}$$

and

$$2^2 = 4 < \sqrt{\frac{17}{7}}$$

so start with

$$\frac{1}{1} < \sqrt{\frac{17}{7}} \quad \frac{2}{1}$$

and do several steps. Of all fractions with denominators up to 10, which are closest to $\sqrt{17/7}$? Using only fractions with denominators up to 10, give a low estimate of $\sqrt{17/7}$ and a high estimate. If your low estimate is used as a value for $\sqrt{17/7}$, what is the maximum possible error?

As you can see, this method of approximating square roots is a simple repetitive procedure, just the right kind of thing for the computer. The first thing the computer must do is find integers that bracket the square root.

Exercise 12-4. Write a program to find integers that bracket the square root of a user-supplied target number TARG. Start with N = 0 and increase N until N*N > TARG. Use your program to find integers that bracket each of the following square roots:

1. $\sqrt{10}$ (answer: $3 < \sqrt{10} < 4$)
2. $\sqrt{17}$
3. $\sqrt{3}$
4. $\sqrt{119}$
5. $\sqrt{2{,}734}$
6. $\sqrt{178{,}653}$

A program to approximate square roots by mediant fractions must first find bracketing integers and then must use fraction operations to compute mediants, square them, and compare the squares to the target fraction. The program MEDIANT ROOT SEARCH (see Listing 12-1) is such a program.

LISTING 12-1

```
100  REM          MEDIANT ROOT SEARCH
110  HOME
120  PRINT "      THIS PROGRAM APPROXIMATES "
130  PRINT : PRINT "THE SQUARE ROOT OF A TARGET FRACTION"
140  PRINT : PRINT "BY STARTING WITH CONSECUTIVE BRACKETING"
150  PRINT : PRINT "INTEGERS."
160  PRINT : PRINT "      IT COMPUTES SUCCESSIVE MEDIANTS"
170  PRINT : PRINT "AND USES THEM AS NEW LOW AND HIGH"
180  PRINT : PRINT "ESTIMATES AS APPROPRIATE."
190  VTAB 23: PRINT "PRESS ANY KEY TO CONTINUE."
200  GET ANS$
210  HOME
220  PRINT : PRINT "ENTER THE TARGET FRACTION."
230  PRINT : PRINT "NUMERATOR";: INPUT NTARG
240  PRINT : PRINT "DENOMINATOR";: INPUT DTARG
250  PRINT : PRINT "MAXIMUM DENOMINATOR FOR BRACKETING"
260  PRINT : PRINT "FRACTIONS";: INPUT BOUND
```

```
270  HOME
280  REM  PRINT HEADER:
290  PRINT : PRINT "APPROXIMATING SQUARE ROOT OF ";NTARG;"/";D
TARG
300  PRINT : PRINT "LOW ESTIMATE";
310  HTAB 20
320  PRINT "HIGH ESTIMATE"
330  POKE 34,4: REM  PROTECT HEADER FROM SCROLLING
340  REM  FIND BRACKETING INTEGERS:
350 N = 0: REM  INITIALIZE N
360  REM  INITIALIZE N2/D2 FOR ORDER SUBROUTINE
370 N2 = NTARG:D2 = DTARG
380  REM   SQUARE N AND SET N1/D1 FOR ORDER SUBROUTINE
390 N1 = N * N:D1 = 1
400  GOSUB 800: REM  COMPARE N*N TO TARGET
410  IF MARK = 1 THEN 470: REM     N*N > TARGET
420  REM  IF N*N < NTARG/DTARG, ADVANCE N:
430  IF MARK =  - 1 THEN N = N + 1: GOTO 380
440  REM   HERE, MARK = 0, SO N*N = NTARG/DTARG
450  PRINT : PRINT "PERFECT SQUARE!"
460  PRINT : PRINT N;"*";N;" = ";NTARG;"/";DTARG: POKE 34,0:
END
470  REM  INITIALIZE BRACKETING INTEGERS LOW AND HIGH:
480 LOW = N - 1:HIGH = N
490  REM  INITIALIZE TARGET FOR ORDER SUBROUTINE
500 N2 = NTARG:D2 = DTARG
510  REM  INITIALIZE BRACKETING FRACTIONS
520 NL = LOW:DL = 1:NH = HIGH:DH = 1
530  REM  PRINT CURRENT BRACKET:
540  PRINT : PRINT NL;"/";DL;
550  HTAB 20
560  PRINT NH;"/";DH
570  REM  COMPUTE MEDIANT OF BRACKETING FRACTIONS:
580 NM = NL + NH:DM = DL + DH
590  REM  SEE IF DENOMINATOR HAS EXCEEDED BOUND:
600  IF DM > BOUND THEN 740: REM  END
610  REM  SQUARE MEDIANT AND PREPARE FOR ORDER SUBROUTINE
620 N1 = NM * NM:D1 = DM * DM
630  GOSUB 800: REM    COMPARE MEDIANT AND TARGET IN ORDER SUB
ROUTINE
640  REM   IF SQUARE OF MEDIANT < TARGET,
650  REM     MAKE MEDIANT NEW LOW BOUND
660  IF MARK < 0 THEN NL = NM:DL = DM: GOTO 530
670  REM  IF SQUARE OF MEDIANT > TARGET,
680  REM     MAKE MEDIANT NEW HIGH BOUND
690  IF MARK > 0 THEN NH = NM:DH = DM: GOTO 530
700  REM   HERE, MARK = 0, SO SQUARE OF MEDIANT = TARGET
710  PRINT : PRINT "PERFECT SQUARE!"
720  PRINT : PRINT "SQUARE OF ";NM;"/";DM;" = ";NTARG;"/";DTAR
G
730  GOTO 770: REM  END
740  REM  ENDING CEREMONY:
750  PRINT : PRINT "THIS IS THE CLOSEST BRACKETING"
760  PRINT : PRINT "WITH DENOMINATORS UP TO ";BOUND
770  POKE 34,0: REM  END SCROLLING PROTECTION
780  END
790  REM
800  REM  SUBROUTINE TO ORDER FRACTIONS
810  REM  N1/D1 AND N2/D2 IN
820  REM  RETURNS:
830  REM                     MARK = -1 IF N1/D1<N2/D2
840  REM                    MARK = 0 IF N1/D1=N2/D2
850  REM                    MARK = 1 IF N1/D1>N2/D2
860  REM  CROSS MULTIPLY:
870 A = N1 * D2:B = N2 * D1
880  IF A < B THEN MARK =  - 1
890  IF A = B THEN MARK = 0
900  IF A > B THEN MARK = 1
910  RETURN
```

Lines 220 to 240 obtain from the operator the fraction NTARG/DTARG whose square root is to be approximated.

Lines 250 to 260 ask for the maximum denominator to be allowed in the computations.

Lines 340 to 480 find integers that bracket the square root. Line 400 calls the ORDER subroutine (lines 800 to 910) to compare the square of an integer to the target fraction.

Line 520 initializes the bracketing fractions NL/DL and NH/DH.

Lines 530 to 560 print the current bracket.

Lines 570 to 580 compute the mediant of the bracketing fractions.

Lines 590 to 600 see whether the maximum denominator has been exceeded. If so, the program ends.

If the maximum denominator has not been reached, the program proceeds in lines 610 to 620 to compute the square of the mediant.

Line 630 calls the ORDER subroutine to compare the square of the mediant to the target fraction.

If the square of the mediant is less than the target, line 660 makes the mediant the new low estimate. Then the program recycles to the printing of the new bracket.

If the square of the mediant is larger than the target, line 690 makes the mediant the new high estimate and recycles.

If the square of the mediant equals the target, lines 710 to 720 announce that happy fact, and the program ends.

Exercise 12-5. Run MEDIANT ROOT SEARCH to see what it does. Approximate the square root of 10 with denominators up to 30.

Exercise 12-6. Insert a line to make the program pause after each bracket printing. Run this pausing version to approximte the square root of $\frac{1}{2}$ with denominators up to 100. Check that each new estimate is the mediant of the old estimates.

Exercise 12-7. Approximate the square root of 2 with denominators up to 200. Compare the decimal representations of the bracketing fractions to the computer's SQR(2).

Exercise 12-8. Modify MEDIANT ROOT SEARCH so that it prints the maximum possible error for the final estimates. (To speed things up, you may also wish to eliminate the pause inserted in Exercise 12-6.) Approximate the square root of $^{45}/_{67}$ by fractions with denominators up to 1,000. The answer is

$$\frac{495}{604} > \sqrt{\frac{45}{67}} < \frac{604}{737}$$

and the maximum error is $1/445{,}148$. As you can see, this method quickly gets rather accurate approximations.

Exercise 12-9. Approximate each of the following square roots. Use denominators up to 1,000

1. $\sqrt{37}$
2. $\sqrt{144/169}$
3. $\sqrt{2}$
4. $\sqrt{3}$
5. $\sqrt{6{,}787}$

Exercise 12-10. Among all fractions with denominators up to 1,500, find the smallest one that is larger than $\sqrt{17}$.

Exercise 12-11. Among all fractions with denominators up to 350, find the largest one that is smaller than $\sqrt{10}$.

Exercise 12-12. Find a fraction that is between $\sqrt{167}$ and $\sqrt{168}$.

In the following exercises, no use of decimals is allowed.

Exercise 12-13. Back in Idaho, Mary had a vegetable garden which was 37 feet wide and 69 feet long. Now that she has moved to Missouri, she wants a garden with the same area, but square this time. With her yard stick, she can measure any fraction of a foot down to 1 inch = $1/12$ foot. What side length (in feet) should she use for her new square garden? Give her two choices—one for a garden that is slightly too large, and one that is slightly too small.

Exercise 12-14. While you were doing Exercise 12-13, Mary went out and bought a new tape measure. Now she can measure any fraction of a foot down to 1/16 inch. With this new accuracy available, what dimensions could she use?

Exercise 12-15. Cleopatra has a circular rose garden. The length of its radius is 11 asps. Antony wants such a garden too, but Cleo doesn't care for competition. So she says Antony can make a garden if its area is no more than one-third the area of hers. What radius should Antony use to get as large a garden as possible? Give the answer as a fractional number of asps. Antony can't handle fractions very well, so give the answer as a fraction with denominator no larger than 20.

Exercise 12-16. The pharaoh's hallway needs some repairs. A ladder is to be used to get up to a ledge that is 1 cubit wide and 6 cubits off the floor. The floor is slippery, so the ladder must be braced against the opposite wall, 10 cubits away. (See Figure 12-2.) The ladder must be long enough to rest on the ledge (not on the painting on the lower wall) and short enough to miss the painting on the upper wall. What length of ladder should be used? Give the answer as a fraction with a denominator less than 30.

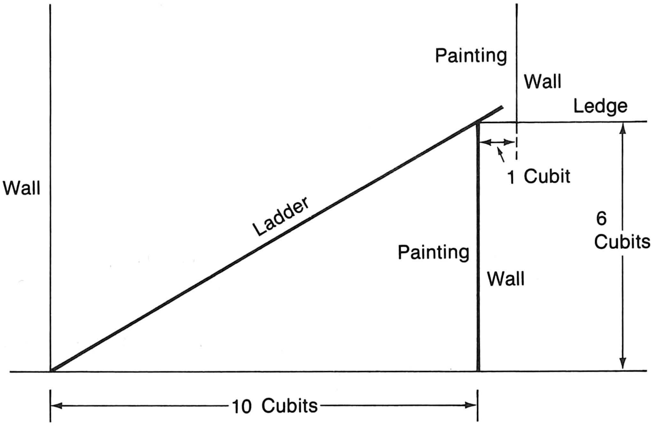

Painting
Wall
Ledge
1 Cubit
Wall
Ladder
6 Cubits
Painting
Wall
10 Cubits

Figure 12-2 Pharaoh's hallway

13

DECIMALS AND LONG DIVISION

After giving fractions a lot of attention, we now give some attention to decimals. Decimals are in fact a shorthand notation for fractions. For example,

$$0.12 = \frac{12}{100}$$

$$0.345 = \frac{345}{1000}$$

There are two good reasons why we often use decimals rather than fractions: First, they are easier to add. For example, it is easier to compute the sum $3/16 + 4/25$ by adding their decimal equivalents $0.1875 + 0.16$, than it is to convert them to fractions with a common denominator, add the numerators, and then reduce to lowest terms.

Second, decimals are easier to compare than fractions. For example, it is easier to see that $4/25 < 3/16$ by looking at their decimal equivalents $0.16 < 0.1875$, than it is to cross-multiply in the fraction inequality and compare the results.

On the other hand, decimals have a serious defect: Many fractions do not have finite decimal equivalents. For example,

$$\frac{1}{3} = 0.3333333\ldots$$

$$\frac{3}{7} = 0.4285714285714\ldots$$

Thus any time we use division in a decimal computation, we must be prepared to round off the answer. This continual rounding-off introduces errors and can have surprising and disastrous results. (See Chapter 20, ‘‘Linear Equations and Round-Off Error.’’)

By now you are accustomed to fractions with infinite decimal expansions, but do you really know what it means to say that a fraction such as 3⁄7 is equal to an infinite string of numbers? In addition, if we can make sense of that, how do we find the digits in the decimal expansion?

You know that the answer to the second question is long division. However, we start here with the first question, which has to do with bracketing fractions, and see how bracketing fractions leads to long division.

We start by discussing the expansion of 3⁄7. The infinite decimal expansion for 3⁄7 really stands for a sequence of bracketings of 3⁄7. The bracketing

$$\frac{4}{10} \le \frac{3}{7} < \frac{5}{10}$$

implies that the first digit in the decimal expansion of 3⁄7 is 4. The bracketing

$$\frac{42}{100} \le \frac{3}{7} < \frac{43}{100}$$

shows that the first two digits in the decimal expansion of 3⁄7 are 4 and 2.

The first three digits, 4, 2, and 8, in the expansion of 3⁄7 are given by the bracketing

$$\frac{428}{1{,}000} \le \frac{3}{7} < \frac{429}{1{,}000}$$

Bracketings with ever higher powers of 10 as denominators give further digits in the expansion. Hence, an infinite decimal expansion for a fraction is actually an infinite sequence of bracketings by fractions with powers of 10 as denominators.

Now that we have seen what an infinite decimal expansion means, let’s see how to find the digits in the expansion. It is easy to make the computer find bracketings, and from these we can read off the digits. The program FRACTION BRACKETER (see Listing 13-1) finds bracketings by fractions with any user-supplied denominator, not just powers of 10.

LISTING 13-1

```
100  REM          FRACTION BRACKETER
110  HOME
120  PRINT "       THIS PROGRAM FINDS FRACTIONS"
130  PRINT : PRINT "WHICH BRACKET A USER-SUPPLIED"
140  PRINT : PRINT "FRACTION."
150  PRINT : PRINT "     YOU SUPPLY THE DENOMINATOR FOR "
160  PRINT : PRINT "THE BRACKETING FRACTIONS."
170  VTAB 23
180  PRINT "       PRESS ANY KEY TO CONTINUE."
190  GET ANS$
200  HOME
210  PRINT "       ENTER THE FRACTION A/B WHICH"
220  PRINT : PRINT "YOU WANT TO BRACKET."
```

```
230   PRINT : PRINT "      A = ";: INPUT A
240   PRINT : PRINT "      B = ";: INPUT B
250   HOME
260   PRINT : PRINT "      WHAT DENOMINATOR DO YOU WANT"
270   PRINT : PRINT "DO YOU WANT TO USE FOR THE "
280   PRINT : PRINT "BRACKETING FRACTIONS";: INPUT DEN
290   HOME
300   PRINT : PRINT "WE WILL SEARCH FOR FRACTIONS"
310   PRINT : PRINT "      NUM/";DEN;" AND (NUM+1)/";DEN
320   PRINT : PRINT "SUCH THAT"
330   PRINT : PRINT "      NUM/";DEN;" <= ";A;"/";B;" < (NUM+1)/
";DEN
340   PRINT : PRINT "WHAT NUMERATOR <NUM> DO YOU WANT"
350   PRINT : PRINT "TO START WITH?"
360   PRINT : PRINT "TRY TO CHOOSE <NUM> SO THAT"
370   PRINT : PRINT "      NUM/";DEN;" <= ";A;"/";B
380   PRINT : PRINT "NUM = ";: INPUT NUM
390   HOME
400   REM   SEE IF NUM/DEN <= A/B
410   IF NUM * B <  = A * DEN THEN 460
420   REM  HERE, STARTING NUM/DEN WAS TOO BIG
430   PRINT : PRINT "YOU GOOFED!"
440   PRINT : PRINT A;"/";B;" < ";NUM;"/";DEN
450   GOTO 570: REM  END
460   REM  INITIALIZE Q:
470   LET Q = 0
480   REM   SEE IF DESIRED Q HAS BEEN FOUND
490   REM  THAT IS, SEE IF A/B < (NUM+Q+1)/DEN
500   IF A * DEN < B * (NUM + Q + 1) THEN 550: REM  END
510   REM    HERE, Q NOT YET LARGE ENOUGH
520 Q = Q + 1
530   PRINT NUM + Q;"/";DEN;" <= ";A;"/";B
540   GOTO 480
550   REM     HERE, HAVE FOUND DESIRED Q
560   PRINT NUM + Q;"/";DEN;" <= ";A;"/";B;" < ";NUM + Q + 1;"/
";DEN
570   PRINT : PRINT "DO ANOTHER ONE (Y OR N)";: INPUT ANS$
580   IF ANS$ = "N" THEN  END
590   GOTO 200
```

We use this program now to get some experience finding bracketings.

Exercise 13-1. Run FRACTION BRACKETER to see how it works.

Now analyze the program:

Lines 100 to 380 obtain the fraction A/B to be bracketed, the denominator DEN to be used, and the starting numerator NUM.

Line 410 tests whether the starting fraction NUM/DEN is already larger than A/B. If so, the program ends. If not, that is, if NUM/DEN ≤ A/B, the program searches for Q, such that

$$\frac{\text{NUM} + \text{Q}}{\text{DEN}} \leq \frac{\text{A}}{\text{B}} < \frac{\text{NUM} + \text{Q} + 1}{\text{DEN}}$$

Lines 480 to 540 repeatedly advance Q until such a Q is found.

Finally, line 560 prints out the bracketing.

Exercise 13-2. Bracket 1/7 by fractions with denominator equal to

1. 17 (Answer: $2/17 \leq 1/7 < 3/17$)
2. 56
3. 33
4. 10
5. 100
6. 1,000

From your result in part 6, what are the first three digits in the decimal expansion of $1/7$?

Exercise 13-3. Bracket $7/17$ by fractions with denominator equal to

1. 5
2. 7
3. 49
4. 10
5. 100
6. 1,000

From your result in part 6, what are the first three digits in the decimal expansion of 7/17?

FRACTION BRACKETER can take a long time when working with a large denominator, such as 1,000. In order to find the digits in the decimal expansion quickly, we need a more efficient method for finding the bracketings. For the power of 10 denominators 10, 100, 1,000, . . . , you can save time by using the information obtained at one stage to help in choosing the starting numerator for the next stage. For example, for the fraction 1,234/5,678, denominator 10 gives the bracketing

$$\frac{2}{10} \leq \frac{1{,}234}{5{,}678} < \frac{3}{10}$$

This implies that $20/100 \leq 1{,}234/5{,}678$. Thus, for denominator 100, you can start with numerator 20 rather than 1. This gives the bracketing

$$\frac{21}{100} \leq \frac{1{,}234}{5{,}678} < \frac{22}{100}$$

This implies that $210/1{,}000 \leq 1{,}234/5{,}678$. Thus, for denominator 1,000, you can start with numerator 210. This gives the bracketing

$$\frac{217}{1{,}000} \leq \frac{1{,}234}{5{,}678} < \frac{218}{1{,}000}$$

Thus, for denominator 10,000, you can take the starting numerator to be 2,170.

Exercise 13-4. Use the method above of choosing starting numerators to bracket $1{,}234/5{,}678$ with fractions having denominator 1,000. Compare the time it takes this way with the time it takes if you start from scratch with denominator 1,000 and numerator 0.

Exercise 13-5. Use this more efficient method to bracket each of the following by fractions having denominator 10,000:

1. $5/17$
2. $34/59$
3. $7689/3051$
4. $2/6$

The procedure above for going from a bracketing for one power of 10 denominator to the next can be done by the computer. Specifically, to go from one stage to the next, multiply the old numerator and denominator by 10 and then restart the bracket search. The next few exercises show how to modify FRACTION BRACKETER so that it uses powers of 10 as denominators and goes from one stage to the next automatically.

Exercise 13-6. Delete lines

```
150 PRINT: PRINT "YOU SUPPLY THE DENOMINATOR FOR"
160 PRINT: PRINT "THE BRACKETING FRACTIONS. "
```

Exercise 13-7. Delete lines 260 to 390, which obtain the starting values for NUM and DEN.

Exercise 13-8. Insert lines

```
260 NUM = 0
270 DEN = 1
```

to initialize NUM and DEN.

Exercise 13-9. Delete line

```
590 GO TO 200
```

Exercise 13-10. Insert lines to make the new numerator and denominator 10 times the old ones:

```
590 NUM = (NUM + Q) *10
600 DEN = DEN*10
```

Exercise 13-11. After getting the new numerator and denominator, we want to search for the new bracketing, so insert line

```
610 GO TO 460
```

Exercise 13-12. Use this version of FRACTION BRACKETER to bracket the following fractions. Find bracketing fractions with denominators equal to 100,000.

1. $4/7$
2. $9/17$
3. $345/567$
4. $5/19$

Exercise 13-13. Use this program to bracket the fractions $2/3$ and $2/7$. Use ever larger and larger denominators. What happens? Why does this happen?

As you can see by the results of Exercise 13-13, this program can give only a few digits in the decimal expansion of a fraction. This is because the program carries along and prints the entire numerator NUM and denominator DEN for each bracketing fraction. NUM and DEN get large very quickly and eventually are so large that the addition NUM + Q + 1 and the comparison in line 500 cannot be carried out correctly.

To begin improving the program in order to get more digits of the expansion, modify it so that it prints just the digits Q of the decimal expansion.

Exercise 13-14. Delete line 530

```
530 PRINT NUM+Q; "/"; DEN; " <= "; A; "/"; B
```

Change line 560 to

```
560 PRINT: PRINT "NEXT DIGIT IS "; Q
```

Exercise 13-15. Use this version to find the first few digits in the decimal expansions of

1. $2/3$
2. $1/4$
3. $1/7$
4. $23/77$

Note that this version is still limited to providing only a few digits correctly. This is because the program still works with NUM and DEN. As a first step toward eliminating the computing of NUM and DEN, consider line 500, where the program decides whether the next digit Q has been found:

```
500 IF A*DEN < B* (NUM+Q+1) THEN 550: REM END
```

The inequality here is

$$A*DEN < B*(NUM + Q + 1)$$

When the right-hand side is multiplied out, this becomes

$$A*DEN < B*NUM + B*(Q + 1)$$

Subtracting B*NUM from both sides gives

$$A*DEN - B*NUM < B *(Q + 1)$$

If we let

$$DN = A*DEN - B*NUM$$

this inequality becomes

$$DN < B*(Q + 1)$$

We now incorporate this into the program.

Exercise 13-16. Insert line 475

```
475 DN = A*DEN-B*NUM
```

Delete line 490

```
490 REM THAT IS, SEE IF A/B < (NUM+Q+1)/DEN
```

Change line 500 so that it uses the new inequality

```
500 IF DN < B*(Q+1) THEN 550
```

Exercise 13-17. Run this version for various fractions to see that it gives correct digits out as far as the previous version did. Note also that it resorts to giving zeros after a while, just as the previous version did.

To eliminate completely the computing of NUM and DEN, look at how DN changes from one stage to the next. We have

$$DN = A*DEN - B*NUM$$

and for the next stage, lines 590 and 600 set

$$\text{NUM} = (\text{NUM} + \text{Q})*10$$
$$\text{DEN} = \text{DEN}*10$$

Hence, the new DN is

$$\text{new DN} = \text{A}*\text{DEN}*10 - \text{B}*(\text{NUM} + \text{Q})*10$$

Factoring out the 10 on the right-hand side gives

$$\text{new DN} = (\text{A}*\text{DEN} - \text{B}*\text{NUM} - \text{B}*\text{Q})*10$$

Substituting old DN = A*DEN − B*NUM gives

$$\text{new DN} = (\text{old DN} - \text{B}*\text{Q})*10$$

Thus, if we let R = DN − B*Q at the end of one stage, the new DN is R*10. Using this R allows us to eliminate the computing of NUM and DEN.

Exercise 13-18. The old lines 590 and 600 are

```
590 NUM = (NUM+Q) *10
600 DEN = DEN*10
```

Change these lines to

```
590 R = DN-B*Q
600 DN = R*10
```

The variables NUM and DEN still appear in line 475, though, as follows:

```
475 DN = A*DEN-B*NUM
```

Since we now have lines 590 and 600 giving the new DN in terms of the old one, we don't need line 475 any more.

Exercise 13-19. Delete line 475

```
475 DN = A*DEN-B*NUM
```

We do, however, need to initialize DN. At the beginning, DEN = 1 and NUM = 0. Thus, at the start,

$$\begin{aligned} DN &= A*DEN - B*NUM \\ &= A*1 - B*0 \\ &= A \end{aligned}$$

Exercise 13-20. Insert line

```
280 DN = A
```

Exercise 13-21. Run this version for various fractions. Note that it does not produce zeros after a few steps as the previous version did.

How does this program avoid the earlier difficulties? Let's look at the algorithm as it now stands. The important lines of the program are

```
280 DN = A
460 REM
470 LET Q = 0
480 REM
500 IF DN < B* (Q+1) THEN 550
520 LET Q = Q+1
540 GO TO 480
550 REM
560 PRINT: PRINT "THE NEXT DIGIT IS ";Q
590 R = DN-B*Q
600 DN = R*10
610 GO TO 460
```

Note that the advancing of Q stops (in line 500) when

$$DN < B*(Q + 1)$$

Multiplying out the right-hand side gives

$$DN < B*Q + B$$

Subtracting B*Q gives

$$DN - B*Q < B$$

But $R = DN - B*Q$, so this says that

$$R < B$$

That is, at the end of each stage, $R < B$. Hence, for the new DN,

$$DN = R*10 < B*10$$

Thus, after the first step when DN = A, DN is always smaller than B*10. Furthermore, R = DN − B*10 is never larger than DN, so neither R nor DN is ever larger than B*10. Thus, the problem of variables getting too large has been avoided.

Also, the program works quickly because the loop 470 to 540 needs at most 10 steps to find the next Q: Since DN < B*10, the first Q that makes

$$\text{DN} < \text{B}*(\text{Q} + 1)$$

is no larger than 9.

To summarize: This algorithm for finding the digits in the decimal expansion produces correct digits forever because the variables R and DN never get large. The algorithm works quickly because it takes no more than 10 steps to get the next digit.

Now let's reinterpret the algorithm. Given DN, it finds Q such that

$$\text{B}*\text{Q} <= \text{DN} < \text{B}*\ (\text{Q} + 1)$$

You have seen this situation before in Chapter 2, "Whole Number Division." DN (*D*ivide*N*d) is divided by B, giving Q (*Q*uotient) and R (*R*emainder), where R = DN − B*Q. Then R is multiplied by 10 and made the dividend for the next round.

This should sound familiar! It's the usual long division algorithm. To see this, compare the long division steps with the program steps in the following example. Find the decimal expansion of $1/7$.

Long Division	Program
Set up diagram $7\overline{)1}$	Initialize variables A = 1, B = 7, DN = A = 1
Divide 1 by 7 $\begin{array}{r} 0. \\ 7\overline{)1.} \\ \underline{0} \\ 1 \end{array}$	Divide 1 by 7 Q = 0, R = 1
Bring down a zero $\begin{array}{r} 0. \\ 7\overline{)1.0} \\ \underline{0} \\ 10 \end{array}$	Get new DN DN = R*10 = 10
Divide 10 by 7 $\begin{array}{r} 0.1 \\ 7\overline{)1.0} \\ \underline{0} \\ 10 \\ \underline{7} \\ 3 \end{array}$	Divide 10 by 7 Q1 = 1, R = 3

Long Division	Program
Bring down a zero	Get new DN
$\begin{array}{r} 0.1 \\ 7\overline{)1.00} \\ \underline{0} \\ 10 \\ \underline{7} \\ 30 \end{array}$	DN = R*10 = 30
Divide 30 by 7	Divide 30 by 7
$\begin{array}{r} 0.14 \\ 7\overline{)1.00} \\ \underline{0} \\ 10 \\ \underline{7} \\ 30 \\ \underline{28} \\ 2 \end{array}$	Q = 4, R = 2

This example shows that the program really does the long division algorithm. Recall that this program evolved from one which found decimal digits from bracketing fractions, so we know that the program gives the correct decimal digits. This shows that the long division algorithm gives the correct digits for the decimal expansion of the fraction, which is not obvious from the long division algorithm itself.

14

PERIODIC DECIMALS

In Chapter 13 we studied infinite decimal expansions of fractions and saw that the long division algorithm generates the digits in the expansions. Now that we have the long division algorithm on the computer, we can quickly compute the decimal expansion of almost any fraction. We use this facility to study some interesting and surprising properties of these expansions. Also, we can easily compute expansions in bases other than 10. Hence, we can complete the discussion of round-off error begun in Chapter 1, ''Stepping.''

Let's start with the program LONG DIVISION (see Listing 14-1).

LISTING 14-1

```
100  REM    LONG DIVISION
110  HOME
120  PRINT "      THIS PROGRAM FINDS THE DECIMAL"
130  PRINT : PRINT "EXPANSION OF FRACTIONS."
140  PRINT : PRINT "     IT USES THE LONG DIVISION ALGORITHM."
150  VTAB 23
160  PRINT "      PRESS ANY KEY TO CONTINUE."
170  GET ANS$
180  HOME
190  PRINT "      TYPE IN THE FRACTION A/B FOR WHICH"
200  PRINT : PRINT "YOU WANT THE DECIMAL EXPANSION."
210  PRINT : PRINT "A = ";
220  INPUT A
230  PRINT : PRINT "B = ";
240  INPUT B
250  HOME
260  PRINT "HIT S TO STOP THE EXPANSION"
270  PRINT "HIT RETURN TO CONTINUE"
280  POKE 34,3: REM  PROTECT FROM SCROLLING
```

```
290  PRINT : PRINT "      THE DECIMAL EXPANSION OF"
300  PRINT : PRINT A;"/";B;" IS "
310  PRINT
320  REM  INITIALIZE FOR ALGORITHM:
330 DN = A
340  REM      COMPUTE DECIMAL DIGIT Q
350  REM   BY DIVIDING DN BY B
360 Q = 0
370  IF DN < B * (Q + 1) THEN 400: REM  QUOTIENT IS Q
380 Q = Q + 1
390  GOTO 370
400  PRINT Q;
410  GET ANS$: REM  PAUSE
420  IF ANS$ = "S" THEN 460: REM  END
430 R = DN - B * Q: REM  REMAINDER
440 DN = R * 10: REM  NEW DIVIDEND
450  GOTO 340
460  POKE 34,0: REM  STOP SCROLL PROTECTION
470  PRINT : PRINT : PRINT "DO YOU WANT TO DO ANOTHER ONE? (Y
OR N)";
480  GET ANS$
490  IF ANS$ <  > "N" THEN 180
500  PRINT : PRINT "PROGRAM STOPPED."
510  END
```

This program uses the long division algorithm to generate the digits in the decimal expansion of fractions.

Lines 210 to 240 obtain the fraction A/B to be expanded.

Lines 330 to 450 contain the long division algorithm in the form we discussed in Chapter 13.

Line 330 initializes the dividend DN.

Lines 360 to 390 divide DN by the denominator B, getting the quotient Q.

Line 400 prints the digit Q.

Lines 410 to 420 give the operator a chance to stop the expansion.

Line 430 computes the remainder R of the previous division.

Line 440 multiplies R by 10 and makes the product the new dividend DN.

Then line 450 starts the next division round.

Exercise 14-1. Run LONG DIVISION for various fractions to see how it works. Try it on $\frac{1}{3}$, $\frac{2}{3}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{4}{3}$, and $\frac{5}{2}$ to see that it gives the correct digits.

Exercise 14-2. Modify LONG DIVISION so that it prints a decimal point. From the expansions you computed in Exercise 14-1, you see that the decimal point should come after the program prints the first Q. In order to print just one decimal point, you can use a variable FLAG. Initialize FLAG by letting FLAG = 0 before starting the expansion of A/B. Print the decimal point if FLAG = 0, and set FLAG = 1 after the decimal point is printed. Run your modified version for various fractions to be sure that it works correctly.

You know from doing long division by hand that sometimes you are lucky and the process terminates. This happens when one of the division steps "comes out even." That is, when a division step gives a remainder R = 0. When this happens, all the rest of the digits Q in the expansion are 0.

Exercise 14-3. Modify your program so that, if a remainder R = 0 at some stage, the program prints a message such as TERMINATED and computes no further digits for the fraction. Try this version on some fractions that you know terminate, such as ¼, ⅛, ⅕. Do you know in general which fractions 1/N have terminating decimals? If so, check some more of them with your program. If not, use your program to try to discover which ones terminate.

As you know, the decimal expansion of every fraction eventually starts to repeat. Why is this? Well, when you do long division on a fraction A/B, the remainder R at each step is required to be smaller than B. Hence, the only possibilities for R are 0, 1, 2, . . . , B − 1. Since there are only a finite number of possibilities for R, the R at one stage will eventually match the R from a previous stage. When that happens, the dividends R*10 and the quotients Q start to repeat.

For example, look at long division applied to 1⁄22:

Step 1:

$$\begin{array}{r} 0 \\ 22\overline{)1} \\ \underline{0} \\ 1 \end{array} \qquad \text{remainder } R = 1$$

Step 2:

$$\begin{array}{r} 0.0 \\ 22\overline{)1.0} \\ \underline{0} \\ 10 \\ \underline{0} \\ 10 \end{array} \qquad \text{remainder } R = 10$$

Step 3:

$$\begin{array}{r} 0.04 \\ 22\overline{)1.00} \\ \underline{0} \\ 10 \\ \underline{0} \\ 100 \\ \underline{88} \\ 12 \end{array} \qquad \text{remainder } R = 12$$

Step 4:

$$
\begin{array}{lll}
 & \overline{0.045} & \\
22) & 1.000 & \\
 & \underline{0} & \\
 & 10 & \\
 & \underline{0} & \\
 & 100 & \\
 & \underline{88} & \\
 & 120 & \\
 & \underline{110} & \\
 & 10 & \text{remainder } R = 10
\end{array}
$$

This remainder, 10, matches the remainder at step 2, so the next division will yield the same digit, 4, as step 3 did. From here on, the digits 4, 5 will repeat in the expansion, so

$$\frac{1}{22} = 0.0454545\ \ldots$$

To save space, we sometimes use a bar to indicate a repeating block:

$$\frac{1}{22} = 0.0454545\ \ldots = 0.0\overline{45}$$

Some fractions, such as

$$\frac{1}{7} = 0.142857142857142857\ \ldots$$

start repeating immediately after the decimal point. Such fractions are called *purely periodic*. Others, such as

$$\frac{1}{22} = 0.0\overline{45}$$

start repeating in some decimal place after the first.

Exercise 14-4. After looking at the expansions of various fractions 1/N, make a guess concerning the form of the numbers N that will give purely periodic fractions 1/N. Then try your guess out on some more fractions, and modify your guess if necessary.

Exercise 14-5. For those numbers N whose fractions 1/N are not purely periodic, figure out how to predict the length of the nonrepeating part.

We now consider the length of the repeating part of a decimal. For example, the decimal expansion of $1/14$ is

$$\frac{1}{14} = 0.07142857142857142 85 \ldots$$

Here the block of digits 714285 is repeated over and over, and this is the shortest block of digits to be repeated. There are 6 digits in this shortest repeating block, so we say that the *period* of the decimal expansion of $1/14$ is 6.

Exercise 14-6. Find the period of $1/7$, $1/3$, $1/11$, $5/9$, $5/17$, and $3/23$.

It can be tedious to find the period of a fraction by scanning the decimal expansion looking for it to start to repeat. (If you're not already convinced of this, try to find the period of $1/59$, $1/89$, or $1/109$.) So, let's make the computer do it for us. As you have seen above, the expansion starts to repeat when the remainder R of one division step matches the remainder from a previous step. We'll have the computer record the remainders of the division steps and then compare each new remainder to the previous ones. When a match is found, we know the repetition has started.

We must add this storing and comparing of remainders to the current long division program. Figure 14-1 is a flow chart for this new part of the program. When the remainder R matches a previous remainder, it is recorded as MAGICR, and the program breaks to a new print line. After R has repeated the first time, the comparing with previous remainders is skipped. The program just checks if R = MAGICR and, if so, breaks to a new print line.

The program LD AUTO BREAK (see Listing 14-2) contains the previous long division program and the new part dealing with the remainders.

LISTING 14-2

```
100  REM         LD AUTO BREAK
102 BOUND = 100
104  DIM R(BOUND): REM  FOR ARRAY OF REMAINDERS
110  HOME
120  PRINT "      THIS PROGRAM FINDS THE DECIMAL"
130  PRINT : PRINT "EXPANSION OF FRACTIONS."
140  PRINT : PRINT "     IT USES THE LONG DIVISION ALGORITHM."
150  VTAB 23
160  PRINT "      PRESS ANY KEY TO CONTINUE."
170  GET ANS$
180  HOME
190  PRINT "      TYPE IN THE FRACTION A/B FOR WHICH"
200  PRINT : PRINT "YOU WANT THE DECIMAL EXPANSION."
210  PRINT : PRINT "A = ";
220  INPUT A
230  PRINT : PRINT "B = ";
240  INPUT B
250  HOME
260  PRINT "HIT S TO STOP THE EXPANSION"
270  PRINT "HIT RETURN TO CONTINUE"
280  POKE 34,3: REM  PROTECT FROM SCROLLING
290  PRINT : PRINT "     THE DECIMAL EXPANSION OF"
```

```
300  PRINT : PRINT A;"/";B;" IS "
310  PRINT
311 COUNTER = 0: REM  INITIALIZE COUNTER
312 SKIP$ = "FALSE": REM  SET = "TRUE" AFTER R REPEATS
314  REM  INITIALIZE ARRAY OF REMAINDERS:
316  FOR K = 1 TO BOUND
318 R(K) = 0
319  NEXT K
320  REM  INITIALIZE FOR ALGORITHM:
330 DN = A
335 FLAG = 0: REM  SET FLAG = 1 AFTER DECIMAL POINT IS PRINTED

340  REM     COMPUTE DECIMAL DIGIT Q
350  REM   BY DIVIDING DN BY B
360 Q = 0
370  IF DN < B * (Q + 1) THEN 400: REM  QUOTIENT IS Q
380 Q = Q + 1
390  GOTO 370
400  PRINT Q;
405  REM  IF THIS WAS FIRST DIVISION, PRINT DECIMAL POINT
407  IF FLAG = 0 THEN  PRINT ".";:FLAG = 1
410  GET ANS$: REM  PAUSE
420  IF ANS$ = "S" THEN 460: REM  END
430 R = DN - B * Q: REM  REMAINDER
440 DN = R * 10: REM  NEW DIVIDEND
450  GOTO 1000: REM  SEE IF R HAS REPEATED
460  POKE 34,0: REM  STOP SCROLL PROTECTION
470  PRINT : PRINT : PRINT "DO YOU WANT TO DO ANOTHER ONE? (Y
OR N)";
480  GET ANS$
490  IF ANS$ <  > "N" THEN 180
500  PRINT : PRINT "PROGRAM STOPPED."
510  END
1000  REM  HAS R ALREADY REPEATED?
1010  IF SKIP$ = "TRUE" THEN 1190: REM  SKIP COMPARING OF R'S
1020  REM  HERE, R HAS NOT YET REPEATED
1030  REM  SEE IF THIS IS FIRST R
1040  IF COUNTER = 0 THEN 1090: REM    NO PREVIOUS R TO COMPAR
E
1050  REM  SEE IF R MATCHES A PREVIOUS R
1060  FOR K = 1 TO COUNTER
1070  IF R = R(K) THEN 1140
1080  NEXT
1090  REM   HERE, NO MATCH WITH PREVIOUS R
1100  REM  ADVANCE COUNTER AND STORE R IN ARRAY:
1110 COUNTER = COUNTER + 1
1120 R(COUNTER) = R
1130  GOTO 340: REM  GET NEXT DIGIT
1140  REM  HERE, R MATCHES PREVIOUS REMAINDER
1150 MAGICR = R: REM  RECORD FIRST REPEATING REMAINDER
1160 SKIP$ = "TRUE": REM  TO SKIP REMAINDER COMPARING
1170  PRINT : REM  BREAK TO NEW LINE
1180  GOTO 340: REM  GET NEXT DIGIT
1190  REM  HERE, MAGICR HAS ALREADY BEEN FOUND
1200  REM  IF R = MAGICR, PRINT ON NEW LINE
1210  IF R = MAGICR THEN  PRINT
1220  GOTO 340: REM   GET NEXT DIGIT Q
```

Let's look at the new lines.

Lines 102 and 104 set the dimension for the array of remainders.

Line 311 initializes COUNTER to 0. COUNTER is used to keep track of the number of remainders that has been stored.

Line 312 initializes a string variable SKIP$ to ‘‘FALSE’’. It is set to ‘‘TRUE’’ after R repeats the first time.

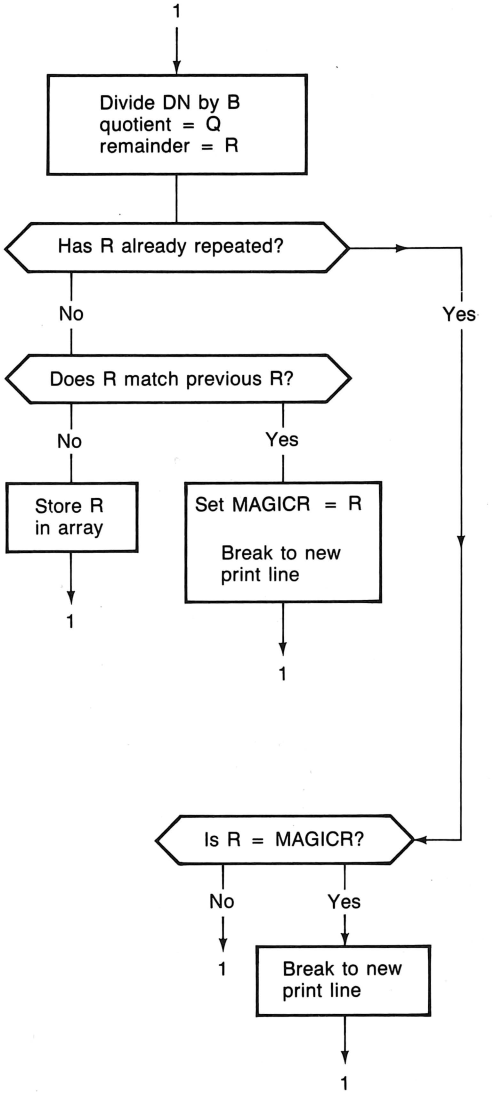

1
Divide DN by B
quotient = Q
remainder = R
Has R already repeated?
No
Yes
Does R match previous R?
No
Yes
Store R in array
1
Set MAGICR = R
Break to new print line
1
Is R = MAGICR?
No
Yes
1
Break to new print line
1

Figure 14-1 Flow chart for remainder comparing

Lines 314 to 319 initialize the array of remainders.

After R and DN are computed, line 450 transfers control to the section that stores and compares remainders, lines 1000 to 1220.

Line 1010 checks whether R has already repeated. If it has, the program determines in line 1210 whether it is time to break to a new print line. Then it recycles to get the next digit of the expansion.

Lines 1030 to 1130 compare R with previous remainders (if any). If no match is found, R is entered in the array, in line 1120. If a match is found, R is recorded as MAGICR in line 1150 and SKIP$ is set to ‘‘TRUE’’ in line 1160. Then the program proceeds to find the next digit.

Exercise 14-7. Run LD AUTO BREAK for various fractions to see that it breaks at the proper places.

Exercise 14-8. Find some fractions with large periods. Note how the printing of the digits slows down as more and more remainders have to be checked. Once the first repeating remainder MAGICR is found, the printing speeds up again.

Exercise 14-9. Compute and write down the expansions of $1/7$, $2/7$, $3/7$, . . . , $6/7$ through at least two repeated blocks of digits. Note that the expansions all contain the same digits in the same order. The only difference is in the starting digit. Does a similar thing happen for other denominators? Look at the expansions of

1. K/14, $1 \le K \le 13$
2. K/13, $1 \le K \le 12$
3. K/19, $1 \le K \le 18$
4. K/31, $1 \le K \le 30$

Exercise 14-10. Look at your expansions for the fractions K/7 again. If you multiply the expansion of $1/7$ by 2, do you get the expansion for $2/7$? What about multiplying the expansion of $3/7$ by 2? Do you get the expansion of $6/7$? What about adding the expansions of $2/7$ and $3/7$? Try similar experiments with other denominators.

Exercise 14-11. The arithmetic operations you investigated in Exercise 14-10 all worked as expected. Now add the expansion of $1/7$ to the expansion of $6/7$. The result is 0.9999999999 . . . . Adding the expansions of $2/7$ and $5/7$ or $3/7$ and $4/7$ result again in the sequence of nines. Try similar additions for other denominators. These results suggest that $1 = 0.999999\ldots$. How can this be? Remember that a decimal expansion stands for a sequence of bracketings by fractions. For example,

$$\frac{3}{7} = 0.428571428571\ldots$$

means

$$\frac{0}{1} \le \frac{3}{7} < \frac{1}{1}$$

$$\frac{4}{10} \le \frac{3}{7} < \frac{5}{10}$$

$$\frac{42}{100} \le \frac{3}{7} < \frac{43}{100}$$

$$\frac{428}{1{,}000} \le \frac{3}{7} < \frac{429}{1{,}000}$$

and so forth.

If we apply this bracketing idea to 1, we get

$$\frac{1}{1} \leq 1 < \frac{2}{1}$$

so the first digit in the expansion of 1 is 1.

$$\frac{10}{10} \leq 1 < \frac{11}{10}$$

so the second digit is 0.

$$\frac{100}{100} \leq 1 < \frac{101}{100}$$

and the third digit is 0.

$$\frac{1{,}000}{1{,}000} \leq 1 < \frac{1{,}001}{1{,}000}$$

and the fourth digit is 0. Thus, we get the expected expansion.

$$1 = 1.00000 \ldots$$

What if we put the $\leq$ on the right in these inequalities rather than on the left? We get equally valid fraction bracketings but possibly different expansions. For $3/7$, we get

$$\frac{0}{1} < \frac{3}{7} \leq \frac{1}{1}$$

$$\frac{4}{10} < \frac{3}{7} \leq \frac{5}{10}$$

$$\frac{42}{100} < \frac{3}{7} \leq \frac{43}{100}$$

and so forth. So these new bracketings give the same old expansion for 3/7. For 1, however, it's quite different:

$$\frac{0}{1} < 1 \leq \frac{1}{1}$$

Thus, the first digit in the expansion of 1 is 0.

$$\frac{9}{10} < 1 \leq \frac{10}{10}$$

so the second digit is 9.

$$\frac{99}{100} < 1 \le \frac{100}{100}$$

so the third digit is 9.

$$\frac{999}{1{,}000} < 1 \le \frac{1{,}000}{1{,}000}$$

so the fourth digit is 9.

Thus, the expansion 0.999999 . . . for 1 is really valid! It results from requiring that the fraction that gives the decimal digits always be smaller than the fraction being expanded.

In terms of long division, this means not allowing any division step to come out even. In other words, we must require a nonzero remainder and hence a (possibly) smaller quotient at each division step. In terms of our program LD AUTO BREAK, it means changing line 370:

```
370 IF DN < B* (Q+1) THEN 400: REM QUOTIENT IS Q
```

We must change the less than sign (<) to a less than or equal sign (≤). To see this, suppose that at line 370 DN equals B*(Q + 1). Then line 370 does not branch to line 400, and so line 380 advances Q to Q + 1. Then line 370 sends control to line 400, and eventually we get R = DN − B*Q in line 430. But this means that R = 0 because this Q is the old Q + 1 and because we assumed that DN = B*(Q + 1) to start with. To avoid this zero remainder, we must make line 370 branch if DN = B*(Q + 1), as we do in Exercise 14-12.

Exercise 14-12. Change line 370 to

```
370 IF DN ≤ B* (Q+1) THEN 400: REM QUOTIENT IS Q
```

Run this version for various fractions whose normal decimal expansions terminate.

Thus you see that some fractions have more than one decimal expansion. For example,

$$\frac{1}{4} = 0.25 = 0.24999999\ldots$$

$$\frac{1}{5} = 0.2 \quad = 0.19999999\ldots$$

Note that this ambiguity only occurs with terminating fractions, so most fractions have unique decimal expansions.

Now let's examine the role that 10, the base of our number system, plays in all of this. We used fractions with powers of 10 as denominators in the bracketings. We could just as well have used denominators that are powers of some other number, say 7.

A terminating base 7 expression such as 0.342 (base 7) stands for

$$\frac{3}{7} + \frac{4}{7^2} + \frac{2}{7^3}$$

A base 7 expansion for a fraction, such as

$$\frac{1}{5} = 0.12541254 \ldots \quad \text{(base 7)}$$

stands for a sequence of bracketings

$$\frac{1}{7} \leq \frac{1}{5} < \frac{2}{7}$$

$$\frac{1}{7} + \frac{2}{7^2} \leq \frac{1}{5} < \frac{1}{7} + \frac{3}{7^2}$$

$$\frac{1}{7} + \frac{2}{7^2} + \frac{5}{7^3} \leq \frac{1}{5} < \frac{1}{7} + \frac{2}{7^2} + \frac{6}{7^3}$$

and so forth.

We know that long division gives the digits in such an expansion, so let's modify our long division program to work with bases other than 10. The only place the base 10 came into our earlier algorithm was in multiplying the remainder at one step to get the dividend for the next step. In the program LD AUTO BREAK, this occurs in line 440:

```
440 DN = R*10: REM NEW DIVIDEND
```

Exercise 14-13. Load LD AUTO BREAK, and change line 440 to

```
440 DN = R*BASE: REM NEW DIVIDEND
```

Exercise 14-14. Insert a line to let you specify the base:

```
245 PRINT: PRINT "WHAT BASE DO YOU WANT TO USE"; : INPUT BASE
```

Exercise 14-15. Change line 290 as follows so that it reports the base being used:

```
290 PRINT: PRINT" THE BASE "; BASE; " EXPANSION OF"
```

Exercise 14-16. Use this version of the program to check the following base 7 expansions:

1. ⅐ = 0.1 (terminates)
2. 1/49 = 0.01 (terminates)

3. $1/343 = 1/(7^3) = 0.001$ (terminates)
4. $1/5 = 0.125412541254 \ldots$

Exercise 14-17. Find the base 3 expansions of

1. $1/3$
2. $1/9$
3. $1/27$
4. $1/2$
5. $1/4$
6. $3/4$

Which of these fractions have terminating base 3 expansions? In general, for what numbers N will 1/N have a terminating base 3 expansion?

You have probably noticed that expansions with a given BASE use the digits 0, 1, . . ., BASE-1. Thus, base 3 expansions use the digits 0, 1, and 2. Base 7 expansions use 0, 1, . . ., 6. As long as the base is no larger than 10, this causes no difficulty with our program. However, for a larger base, such as 16, the expansion may use 0, 1, 2, . . ., 15. With the way our current program prints, we have no way to distinguish a quotient of 15 from the quotient 1 followed by a quotient of 5. To correct this, print a space between each of the quotients Q:

Exercise 14-18. Change line 400 to

```
400 PRINT Q; " ";
```

Exercise 14-19. Use this version of the program to find the base 16 expansions of

1. $1/2$
2. $1/4$
3. $1/16$
4. $1/3$
5. $1/9$
6. $1/11$

Exercise 14-20. As we saw in Chapter 1, "Stepping," the computer works with numbers in base 2 form. Let's look at some base 2 expansions to see how our numbers appear to the machine. Find the base 2 expansions of

1. $1/2$
2. $1/4$

3. $\frac{1}{8}$
4. $\frac{1}{3}$
5. $\frac{1}{5}$
6. $\frac{1}{10}$

For item 6, you get

$$\frac{1}{10} = 0.00011001100110011\ldots \quad \text{(base 2)}$$

which is nonterminating. Thus, when you ask the computer to use $\frac{1}{10}$ in a program, it first converts $\frac{1}{10}$ to base 2. Then, because it can carry only a finite number of digits, it must round off this nonterminating expansion. This generates the round-off error that you observed in Chapter 1, "Stepping."

Exercise 14-21. Compare the base 2 expansions of each of the following pairs of fractions:

1. $\frac{1}{3}$ and $\frac{2}{3}$
2. $\frac{1}{7}$ and $\frac{2}{7}$
3. $\frac{2}{7}$ and $\frac{4}{7}$
4. $\frac{1}{9}$ and $\frac{4}{9}$
5. $\frac{1}{17}$ and $\frac{8}{17}$

Based on what you have seen in these expansions, formulate a rule for multiplying a base 2 expansion by 2, 4, or 8. What is the result of multiplying

$$0.00010011001100110011\ldots \quad \text{(base 2)}$$

by 2? by 4? by 8?

Exercise 14-22. Compare the base 2 expansions of each of the following:

1. $\frac{1}{3}$ and $\frac{2}{3}$
2. $\frac{1}{5}$ and $\frac{4}{5}$
3. $\frac{1}{9}$ and $\frac{8}{9}$

What do you get if you add the expansions of, say $\frac{1}{3}$ and $\frac{2}{3}$? From this example, you see that base 2 expansions have the same ambiguity for terminating expansions as base 10 does.

Exercise 14-23. The ancient Babylonians used a number system based on 60. Why would they have chosen such a huge number for a base? Could 60 have any advantage over our base 10? To investigate this, compute the base 60 expansions for ½, ⅓, ¼, . . ., ⅒ and see how many of them terminate. In our base 10 system, how many of them terminate? Since terminating expansions are easier to compute with, perhaps we should switch to the Babylonian system!

To finish off this chapter, Exercises 14-24 and 14-25 present intriguing questions that you can investigate.

Exercise 14-24. Find the smallest N for which 1/N has period 7.

Exercise 14-25. Find how the period of 1/(M*N) is related to the periods of 1/M and 1/N. (You may wish to concentrate at first on M and N that are primes.)

In attempting to answer these last two exercises, you may want to modify LD AUTO BREAK so that it reports the period of the expansion without actually printing the expansion. When R matches a previous remainder R(K) in line 1070, you can compute the period of the expansion from COUNTER, which records the number of remainders, and K.

15

BRUTE-FORCE PI

In Euclidean geometry, all circles have the same ratio of circumference to diameter. Approximating the value π of this ratio has challenged mathematicians since ancient times. We consider here a simple way to approximate π, and ask you to write a program to do the approximation. We discuss a more sophisticated method in Chapter 16, ''Archimedes' Pi.''

We approximate the circumference of a circle by straight-line segments and then compute the ratio of the approximate circumference to the diameter. This approach is possible because we have a formula for the distance between any two points in the plane.

We use the circle of radius 1 centered at the origin in the X-Y plane. This circle has the equation

$$X^2 + Y^2 = 1$$

and has a diameter $D = 2$. (See Figure 15-1.)

To approximate the circumference, we approximate the length of the first-quadrant arc and then multiply by 4. To approximate the first-quadrant arc length, we start at (0, 1) and take points going toward (1, 0) on the arc. Then we compute the total length of the straight-line segments joining one point to the next along the arc. (See Figure 15-2.)

Here are some suggestions to help you set up a program. To make the choosing of the points easy, use equally spaced X coordinates. If you want to have NUM points after (0, 1), use an X step (S) equal to 1/NUM. Then the first point has X coordinate S, the second 2S, and so on, until $X = 1 = NUM*S$.

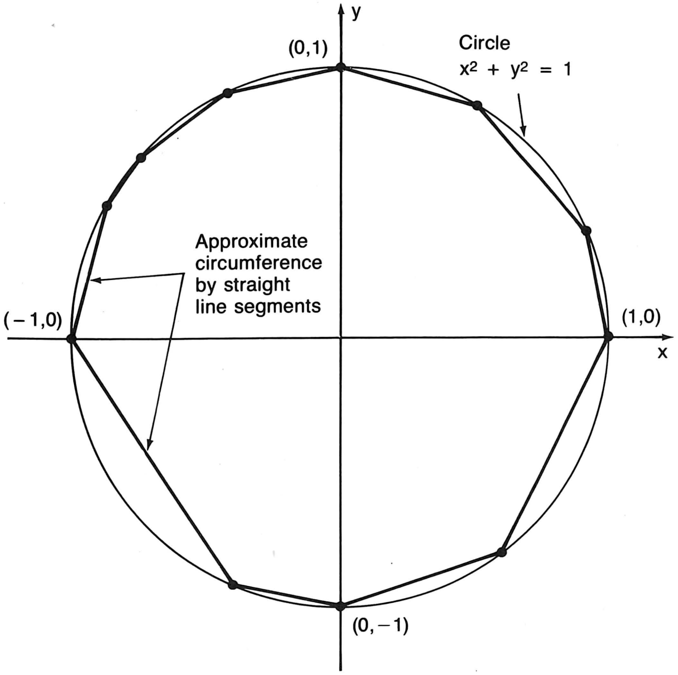

y
(0,1)
Circle
$x^2 + y^2 = 1$
Approximate circumference by straight line segments
(−1,0)
(1,0)
x
(0,−1)

Figure 15-1 Circumference approximation

For each X coordinate, the corresponding Y coordinate is

$$Y = \sqrt{1 - X^2}$$

If (XO, YO) is one point on the arc and (XN, YN) is the next, the length of the segment joining them is

$$LNTH = \sqrt{(XN - XO)^2 + (YN - YO)^2}$$

If ARC is the total length of the line segments up to (XO, YO), update ARC by adding in LNTH:

$$ARC = ARC + LNTH$$

Figure 15-3 is a flow chart for the program.

Exercise 15-1. Write a program to approximate π as outlined above.

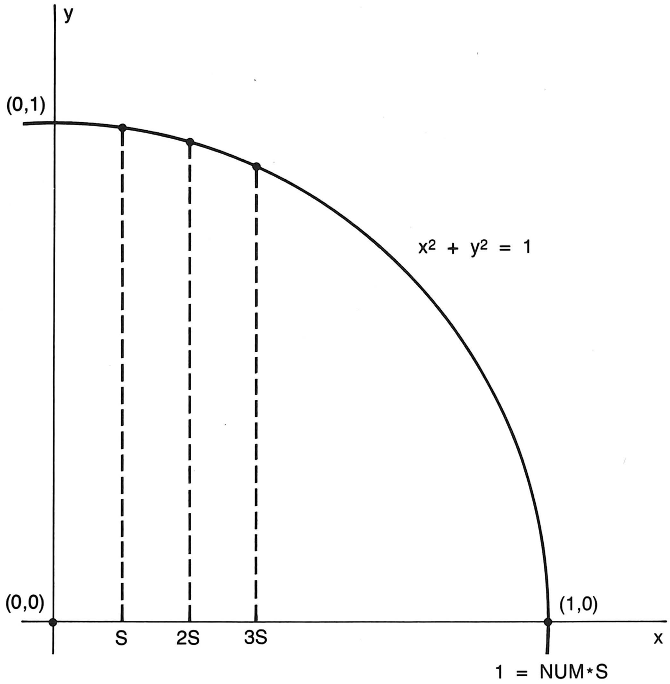


Figure 15-2 First-quadrant arc

Exercise 15-2. For the number of points NUM = 10, 20, 50, 100, and 200, use the program of Exercise 15-1 to approximate π. Record your answers, and compare them with the computer's value of π, which is given by 4*ATN(1).

Exercise 15-3. How many seconds does the program take with 100 points? with 200?

Exercise 15-4. Find how many points are necessary to get an approximation larger than 3.1415.

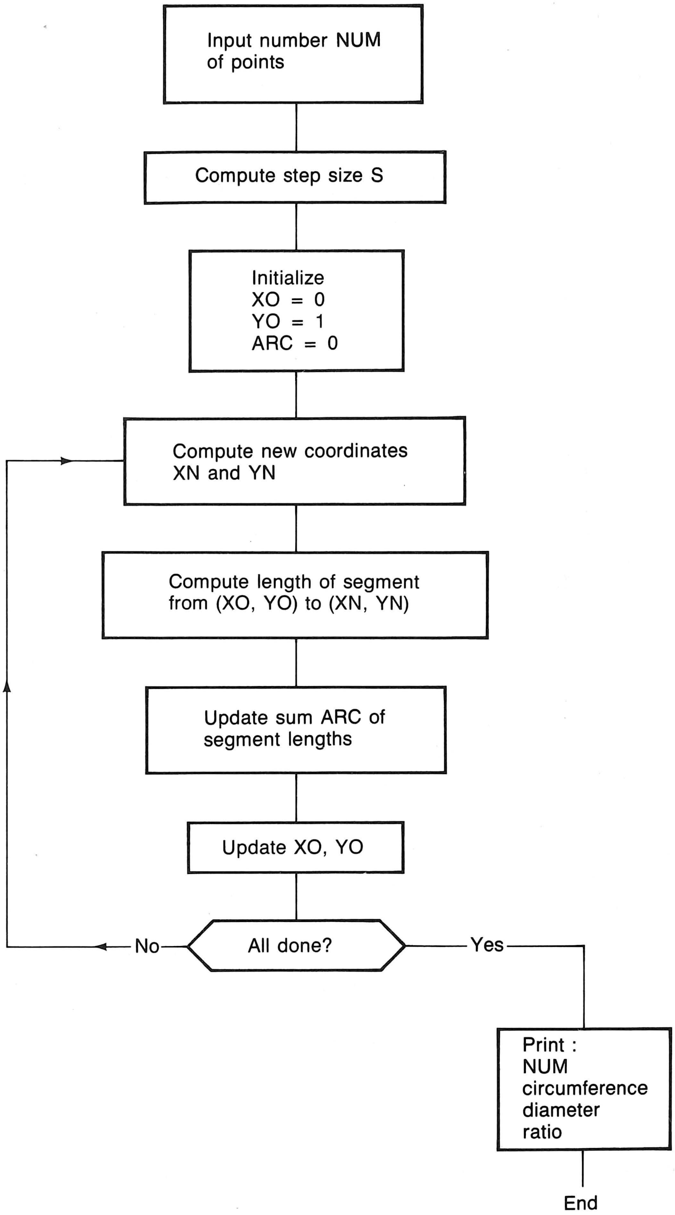

Input number NUM of points
Compute step size S
Initialize
XO = 0
YO = 1
ARC = 0
Compute new coordinates XN and YN
Compute length of segment from (XO, YO) to (XN, YN)
Update sum ARC of segment lengths
Update XO, YO
All done?
No
Yes
Print :
NUM
circumference
diameter
ratio
End

Figure 15-3 Flow chart for BRUTE FORCE PI

16

ARCHIMEDES' PI

In Chapter 15, "Brute-Force Pi," you used a simple but inefficient method for approximating π. We now turn to the method of Archimedes (300 B.C.), which makes much more use of the geometry of the circle. Archimedes' use of geometry results in an amazingly simple algorithm and gives an efficient method of approximating π. The following discussion shows how elementary algebra, geometry, and trigonometry are used to develop the algorithm. In the exercises, you are asked to write a program to approximate π using Archimedes' algorithm.

Start with a circle and circumscribed regular polygon, say a triangle. The perimeter of the triangle may be taken as an approximation to the circumference of the circle. Dividing the perimeter of the triangle by the diameter of the circle gives an approximation to π. Now double the number of sides, giving a hexagon. The perimeter of the hexagon is closer to the circumference of the circle, so a better approximation of π will result. Note that circumscribed polygons have perimeters larger than the circumference, so will give ratios larger than π. Inscribed polygons will give ratios smaller than π.

Continuing to double the number of sides of a circumscribed polygon gives a sequence of ratios, each larger than π, that approaches π. Similarly, repeatedly doubling the number of sides of an inscribed polygon gives a sequence of ratios, each smaller than π, which approaches π.

We first look at the effect of the doubling in the circumscribed case. Start with a circumscribed polygon of S_O sides. (The subscript "O" stands for "old.") Doubling the number of sides gives a polygon of $S_N = 2S_O$ sides. (The "N" stands for "new.") (See Figure 16-1.)

Let V_O be a vertex of the old polygon, V_N a vertex of the new one, and let T be the point of tangency of the line V_OV_N. Then V_OT is half the side of the old circumscribed

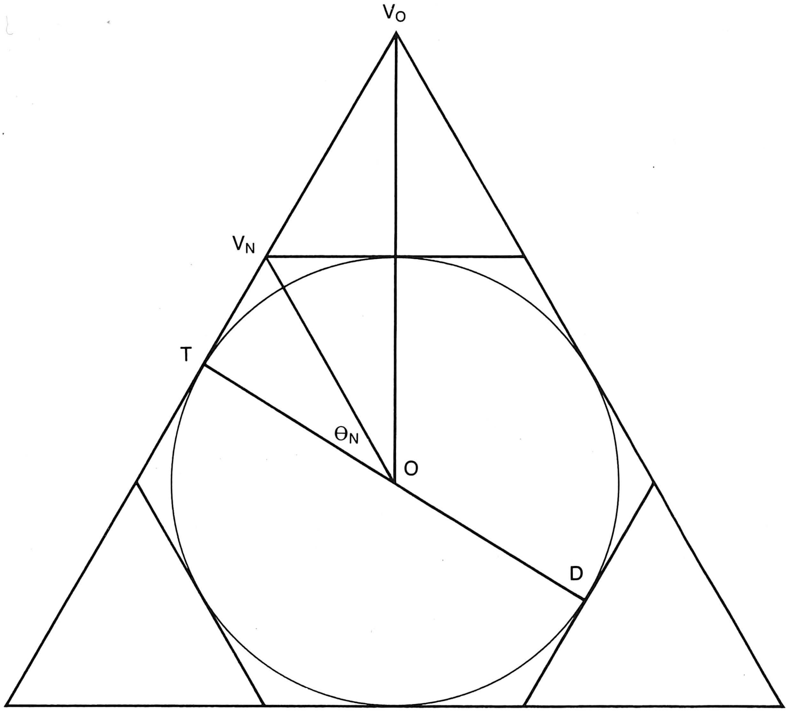


Figure 16-1 Circumscribed polygons

polygon (the triangle in the figure), and V_NT is half the side of the new one (the hexagon). Let TOD be a diameter of the circle. Let θ_N = angle TOV_N and θ_O = angle TOV_O. Then $2\theta_N$ is the central angle in the new polygon, and $2\theta_O$ is the central angle in the old polygon.

The old polygon has S_O sides, each of length $2V_OT$, so the perimeter of the old circumscribed polygon is

$$(CP)_O = S_O(2V_OT)$$

(C denotes "circumscribed," and P denotes "perimeter.")

The new polygon has S_N sides, each of length $2V_NT$, so its perimeter is

$$(CP)_N = S_N(2V_NT)$$

We must find how to compute V_NT in terms of V_OT. To do this, look at the ratio V_OT/V_NT. Because $V_OT = V_OV_N + V_NT$,

$$\frac{V_OT}{V_NT} = \frac{V_OV_N + V_NT}{V_NT}$$

Algebra reduces this to

$$\frac{V_OT}{V_NT} = \frac{V_OV_N}{V_NT} + 1$$

Because OV_N bisects angle TOV_O,

$$\frac{V_OT}{V_NT} = \left(\frac{OV_O}{OT}\right) + 1$$

Dividing through by V_OT gives

$$\frac{1}{V_NT} = \left(\frac{OV_O}{V_OT}\right)\left(\frac{1}{OT}\right) + \frac{1}{V_OT}$$

Then multiplying by OT gives

$$\frac{OT}{V_NT} = \frac{OV_O}{V_OT} + \frac{OT}{V_OT} \qquad (1)$$

This gives a way to compute V_NT in terms of V_OT. If we define

$$(CT)_N = \cot(\theta_N) = \frac{OT}{V_NT}$$

$$(CT)_O = \cot(\theta_O) = \frac{OT}{V_OT}$$

$$(CS)_N = \csc(\theta_N) = \frac{OV_N}{V_NT}$$

$$(CS)_O = \csc(\theta_O) = \frac{OV_O}{V_OT}$$

then equation (1) becomes

$$(CT)_N = (CS)_O + (CT)_O \qquad (2)$$

This tells how to compute the new CT in terms of the old CT and old CS. To complete the transition from old to new, we must also compute the new CS. This can be done by using the identity

$$\csc^2(\theta_N) = 1 + \cot^2(\theta_N)$$

which says

$$(CS_N)^2 = 1 + (CT_N)^2$$

This gives

$$(CS)_N = \sqrt{1 + (CT_N)^2} \tag{3}$$

which shows how to compute the new CS.

Since

$$(CT)_N = \frac{OT}{V_NT}$$

$$V_NT = \frac{OT}{(CT)_N} \tag{4}$$

so we can compute the new VT from the new CT.

Now we can see how to get the upper approximation to π given by the circumscribed polygon. The perimeter of the new circumscribed polygon is

$$\begin{aligned}(CP)_N &= S_N{*}2{*}V_NT \\ &= \frac{S_N{*}2{*}OT}{(CT)_N} \\ &= \frac{S_N{*}TD}{(CT)_N}\end{aligned}$$

and the ratio perimeter/diameter that gives the upper approximation to π is

$$\begin{aligned}\frac{(CP)_N}{TD} &= \frac{S_N}{(CT)_N} \\ &= \text{(number of sides)/(cotangent)}\end{aligned}$$

Before going on to arrange this for the computer, we turn to inscribed polygons to obtain lower approximations to π. Start with an inscribed polygon of S_O sides. Let TOD be a diameter of the circle as before. Let W_O and W_N be vertices of the old and new polygons, respectively. (See Figure 16-2.)

Then angle TOW_N, being a central angle in the inscribed polygon of S_N sides, is equal to $2\ \theta_N$. Hence, angle $TDW_N = \theta_N$, since triangle $W_N\ OD$ is an isosceles triangle and angle $W_NOD = \pi - 2\theta_N$. Also,

$$\begin{aligned}(CS)_N &= \csc(\theta_N) \\ &= \frac{TD}{TW_N}\end{aligned}$$

(TD is the hypotenuse of right triangle TDW_N.) Hence,

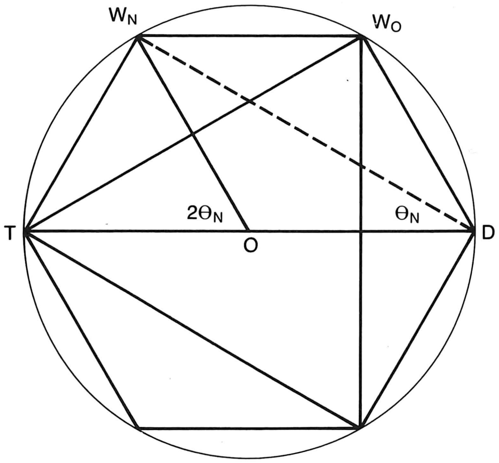


Figure 16-2 Inscribed polygons

$$TW_N = \frac{TD}{(CS)_N} \tag{5}$$

The new inscribed polygon has S_N sides, each of length TW_N, so its perimeter is

$$(IP)_N = S_N{*}TW_N$$

(I stands for "inscribed.") So, by Equation (5),

$$(IP)_N = \frac{S_N{*}TD}{(CS)_N}$$

Thus, the lower approximation to π is

$$\frac{(IP)_N}{TD} = \frac{S_N}{(CS)_N}$$
$$= \text{(number of sides)/(cosecant).}$$

In summary, the upper approximation to π is number of sides/cotangent, and the lower approximation is number of sides/cosecant.

Now it is easy to see how to write a computer program using this algorithm. In computer terms, the procedure is as follows:

1. Initialize:
    a. the number of sides S
    b. the cotangent CT
    c. the cosecant CS

```
2. Print results:
   a. number of sides S
   b. upper π approximation S/CT
   c. lower π approximation S/CS
3. Compute new values:
   a. let CT = CT + CS
   b. let CS = SQR(1 + CT^2)
   c. let S = 2*S
4. Repeat steps 2 and 3 to get the next polygon and its approximations.
```

The outline above for a computer program can be followed immediately except for step 1, the initializing. How do you choose the number S of sides for the starting polygon, and how do you compute the first CT and CS? The angles in this discussion have been half the central angles of the polygons. If our polygon has S sides, its central angle is $2\pi/S$. Thus,

$$\theta = \left(\frac{1}{2}\right)\left(\frac{2\pi}{S}\right) = \frac{\pi}{S}$$

Thus, the initial values are

$$CT = \cot\left(\frac{\pi}{S}\right)$$

$$CS = \csc\left(\frac{\pi}{S}\right)$$

But these expressions already involve π, which we're trying to approximate! So how can we get the algorithm started? Fortunately, the cotangents and cosecants of $\pi/3$, $\pi/4$, and $\pi/6$ can be evaluated geometrically from familiar right triangles. Thus, start with the number of sides S equal to 3, 4, or 6.

Exercise 16-1. Write a program to approximate π using the method shown above. Start with polygons of 3 sides. At each step, print the number of sides, the low estimate for π, and the high estimate of π. How many sides does it take to get within three decimal places of π? Five places? Note: You may see what the computer thinks π is by printing 4*ATN(1).

Exercise 16-2. Repeat Exercise 16-1 for starting number of sides = 4.

Exercise 16-3. Run your program and double the number of sides 20 times. Compare the last decimal place in your approximations with the last place in the computer's 4*ATN(1). When do your approximations get that last digit? Do you notice anything strange in your last few approximations? Whose value of π is better, yours or the computer's?

The method noted above for approximating π computes the contangents and cosecants CT and CS and uses them to compute the perimeters CP and IP. There is a way to compute the new perimeters directly in terms of the old ones, and it happens to use the harmonic and geometric means we have met before. Exercises 16-4 through 16-6 provide an outline and hints for the proof of this method.

Exercise 16-4. Prove that

$$(CP)_N = \frac{2(CP)_O(IP)_O}{(CP_O + IP_O)}$$

This says that the new CP is the harmonic mean of the old CP and the old IP. Hint: Substitute

$$(CP)_O = \frac{S_O * TD}{(CT)_O}$$

$$(IP)_O = \frac{S_O * TD}{(CS)_O}$$

$$(CP)_N = \frac{S_N * TD}{(CT)_N}$$

and use the relations

$$(CT)_N = (CT)_O + (CS)_O$$

and

$$S_N = 2S_O$$

Exercise 16-5. Prove that

$$(CS)_N = \sqrt{2*(CT)_N*(CS)_O}$$

Hint:

$$\begin{aligned} (CS)_N &= \csc(\theta_N) \\ (CT)_N &= \cot(\theta_N) \\ (CS)_O &= \csc(\theta_O) \\ &= \csc(2\theta_N) \end{aligned}$$

so this is an identity to be proved:

$$\csc(\theta) = 2 \cot(\theta) \csc(2\theta)$$

Exercise 16-6. Prove that

$$(\mathrm{IP})_N = \sqrt{(\mathrm{CP})_N(\mathrm{IP})_O}$$

This says that the new IP is the geometric mean of the new CP and the old IP.

Hint: Substitute

$$(\mathrm{IP})_N = \frac{S_N\mathrm{TD}}{(\mathrm{CS})_N}$$

$$(\mathrm{CP})_N = \frac{S_N\mathrm{TD}}{(\mathrm{CT})_N}$$

$$(\mathrm{IP})_O = \frac{S_O\mathrm{TD}}{(\mathrm{CS})_O}$$

and use the result of Exercise 16-5.

These relations allow π to be approximated as follows: Take the circle of diameter 1 so that the perimeters themselves will approximate π. (If the diameter is 1, the ratio perimeter/diameter = perimeter.)

1. Initialize CP and IP to the perimeters of the starting circumscribed and inscribed polygons:
    a. $\mathrm{CP} = S/\cot(\pi/S)$
    b. $\mathrm{IP} = S/\csc(\pi/S)$

2. Print results:
    a. number of sides
    b. upper approximation CP
    c. lower approximation IP
3. Compute new values:
    a. Let $\mathrm{CP} = 2*\mathrm{CP}*\mathrm{IP}/(\mathrm{CP} + \mathrm{IP})$
    b. Let $\mathrm{IP} = \sqrt{\mathrm{CP}*\mathrm{PI}}$
4. Repeat steps 2 and 3 for the next approximations to π.

Exercise 16-7. Write a program to approximate π using the method noted above. Start with polygons of 3 sides. At each step, print the low and high estimates for π. How many sides does it take to get within three decimal places of π? Five places?

Exercise 16-8. Repeat Exercise 16-7 for starting number of sides = 4.

Exercise 16-9. Run your program and double the number of sides 20 times. When do your approximations get the same last digit as the computer's 4*ATN(1)? Note that your approximations behave themselves better using this method. This is because this method does not directly involve the number of sides in the computations.

Now that you have studied the proof of Archimedes' method and seen how it works on the computer, some history may help you appreciate his achievement. Archimedes started with a hexagon and doubled the number of sides four times. His low estimate for π was $3\frac{10}{71}$. His high estimate for π was $3\frac{1}{7}$. He gave his estimates as fractions because he was working some 1800 years before decimals were invented. In fact, he didn't even have a place-value system of numerals! How would you like to do a couple of rounds of his algorithm completely in, say, Roman numerals?

For more history and Archimedes' own proof, see T.L. Heath, ed., *The Works of Archimedes* (Dover Publications Inc., New York.

17

LINEAR EQUATIONS I

Solving systems of linear equations is a standard problem in mathematics. As you know, solving a system of equations can be long and tedious, particularly if there is a large number of equations such as four equations in four unknowns. Moreover, if fractions enter into the arithmetic, you may not have too much confidence in your final answer. Our plan here is to make the computer take over much of the arithmetic and bookkeeping of solving systems of linear equations. We must first see how best to attack such problems with the computer.

You already know a couple of ways to solve a system, such as the following one:

$$\begin{aligned}&(1)\quad 2x + y = 4\\ &(2)\quad x - y = 5\end{aligned}$$

One method is to solve one equation for one of the variables and substitute in the other equation. For example, solving equation (1) for y gives

$$y = 4 - 2x$$

Substituting this in equation (2) gives

$$\begin{aligned}x - (4 - 2x) &= 5\\ x - 4 + 2x &= 5\\ 3x - 4 &= 5\\ 3x &= 9\\ x &= 3\end{aligned}$$

Then, from $y = 4 - 2x$, we get

$$\begin{aligned} y &= 4 - 2*3 \\ y &= -2 \end{aligned}$$

The solution is $x = 3$ and $y = -2$.

This method is not easy to implement on the computer because of the substitution step—plugging $y = 4 - 2x$ into equation (2). The computer works easily with numbers, but not so easily with symbols like 2x.

Another method for solving the system is to manipulate the equations in order to eliminate a variable from one of them. Let's use this method on the same system:

$$\begin{aligned} &(1)\ 2x + y = 4 \\ &(2)\ \ x - y = 5 \end{aligned}$$

Add equation (1) to equation (2), and take the result as a new equation (2). The addition gives us

$$\begin{aligned} 2x + x + y - y &= 4 + 5 \\ 3x + 0 &= 9 \end{aligned}$$

Thus, the new system is

$$\begin{aligned} &(1)\ 2x + y = 4 \\ &(2)\ 3x \phantom{{}+y} = 9 \end{aligned}$$

If we now multiply equation (2) by ⅓, we get

$$\begin{aligned} &(1)\ 2x + y = 4 \\ &(2)\ \ x \phantom{{}+y} = 3 \end{aligned}$$

We have the complete solution if we can eliminate the 2x term from equation (1). To do this, we multiply equation (2) by -2 and add it to equation (1), as follows. Multiplying equation (2) by -2, we obtain

$$\begin{aligned} &(1)\ \ \ 2x + y = 4 \\ &(2)\ -2x \phantom{{}+y} = -6 \end{aligned}$$

Adding equation (2) to equation (1), we get

$$\begin{aligned} &(1)\ \phantom{-2x+{}} y = -2 \\ &(2)\ -2x \phantom{{}+y} = -6 \end{aligned}$$

Now we restore equation (2) by multiplying it by −½:

$$(1)\ y = -2$$

$$(2)\ x = 3$$

So we again have the solution to the system.

This method may seem longer than the first one, but it has a distinct advantage as far as the computer is concerned. It operates only on the constant terms and the coefficients of the variables. To make the computer do these operations, we need only keep track of which numbers are coefficients of which variables and which numbers are constants for which equations. We can do this easily with arrays.

One change which we want to make from what you are accustomed to is in notation. We want to make a change in notation so that we are able to solve systems of equations with any number of equations and variables. If we continue to use the standard symbols x, y, z, and so forth for variables, we will run out of symbols. Hence, we need a better notation for the variables. We will use X_1, for the first variable, X_2 for the second, and so on. In this notation, we would write the system we discussed above as

$$(1)\ 2X_1 + X_2 = 4$$

$$(2)\ X_1 - X_2 = 5$$

Another perhaps unexpected arrangement we must make is in the way we handle the numbers involved as coefficients and constants. We must deal extensively with multiplicative inverses (note the multiplying by ⅓ and −½ above). If we try to work with decimals here, round-off error causes considerable difficulty (see Chapter 20). To avoid this problem, we work exclusively with fractions.

In the remainder of this chapter, we help you develop a program to solve a system of linear equations by the method outlined above. By the end of the chapter, you will have a program with which you can solve a system of (almost) any number of equations and variables.

We will need two-dimensional arrays to hold the numerators and denominators of the coefficients and constants. For the arithmetic operations, we use the fraction subroutines which you met in Chapter 8. To add one equation to another, we use the fraction adding subroutine to combine corresponding coefficients. To multiply an equation by a constant, we use the fraction-multiplying subroutine.

First we need to discuss the notation used in the program. We will work with linear equations in several variables, such as

$$\left(\frac{3}{4}\right) X_1 + \left(\frac{2}{5}\right) X_2 + \left(\frac{5}{6}\right) X_3 = -6 \qquad (17\text{-}1)$$

$$4X_1 + -2\ X_2 + 3X_3 = \quad 8 \qquad (17\text{-}2)$$

VARS will be the number of variables, and EQNS will be the number of equations. In the example above, VARS = 3 and EQNS = 2.

A variable ROW is used to designate an equation, and COL designates a variable or constant term. (See Figure 17-1.)

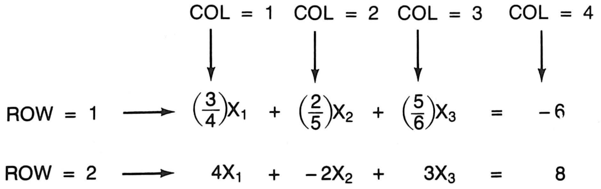


Figure 17-1 COL and ROW diagram

We use the following notation for the fractions appearing as coefficients or constants. N(ROW, COL)/D(ROW, COL) is the coefficient of X_{COL} in equation ROW. In equation (17-1) above,

¾ is the coefficient of X_1, so N(1, 1) = 3 and D(1, 1) = 4.
⅖ is the coefficient of X_2, so N(1, 2) = 2 and D(1, 2) = 5.
⅚ is the coefficient of X_3, so N(1,3) = 5 and D(1, 3) = 6.
−6 is the constant term in equation (17-1), so N(1, 4) = −6 and D(1, 4) = 1.

In general, for equation ROW, N(ROW, COL)/D(ROW,COL) is the coefficient of X_{COL}, for $1 \leq$ COL $\leq$ VARS, and N(ROW, VARS + 1)/D(ROW,VARS + 1) is the constant term.

To begin on a program to work with systems of equations, consider the program EQUATION GETTER. This contains the fraction subroutines and a subroutine to get a system of equations from the user. Run EQUATION GETTER to see the starting locations of the fraction subroutines. The equation-getting subroutine follows

LISTING 17-1

```
1070  REM  SUBROUTINE TO GET EQUATIONS
1080  REM
1090  HOME
1100  PRINT "HOW MANY VARIABLES";: INPUT VARS
1110  PRINT : PRINT "HOW MANY EQUATIONS";: INPUT EQNS
1120  DIM N(EQNS,VARS + 1),D(EQNS,VARS + 1)
1130  FOR ROW = 1 TO EQNS
1140  HOME
1150  REM  GET EQUATION "ROW":
1160  REM
1170  PRINT "TYPE IN  A(1)...A(";VARS;"),";
1180  PRINT " AND B ": PRINT : PRINT "FOR EQUATION ";ROW: PRINT

1190  FOR COL = 1 TO VARS
1200  IF COL < VARS THEN  PRINT "A(";COL;")X";COL;"+";
1210  IF COL = VARS THEN  PRINT "A(";COL;")X";COL;"=";
1220  NEXT COL
1230  PRINT "B";: PRINT
1240  FOR COL = 1 TO VARS
1250  PRINT : PRINT "A(";COL;")=";: GOSUB 820: REM   FRAC GETT
ER
```

```
1260 N(ROW,COL) = NUM:D(ROW,COL) = DEN
1270  NEXT COL
1280  PRINT : PRINT "B";"=";: GOSUB 820: REM   FRAC GETTER
1290 N(ROW,VARS + 1) = NUM:D(ROW,VARS + 1) = DEN
1300  PRINT : PRINT "CHECK THE EQUATION": PRINT
1310  GOSUB 1400: REM  PRINT EQUATION ROW
1320  PRINT : PRINT "IS THIS CORRECT (Y OR N)";
1330  REM
1340  INPUT ANS$: IF ANS$ = "N" THEN  HOME : GOTO 1170
1350  IF ANS$ <  > "Y" THEN 1340
1360  NEXT ROW
1370  REM    PRINT ALL EQUATIONS:
1380  GOSUB 2600
1390  RETURN
1400  REM
```

Now let's analyze the equation-getting subroutine so you can see how the arrays are used.

Lines 1100 to 1110 obtain the number of variables and equations.

Line 1120 sets the dimensions of the arrays that hold the numerators and denominators of the coefficients and constants.

Lines 1170 to 1230 show the notation to be used in asking for the coefficients and constants.

Exercise 17-1. To see these introductory messages,

1. Insert a line:

```
1235 END
```

2. Type RUN 1070 and specify, for instance, 2 variables and 2 equations.
3. Remove line:

```
1235 END
```

Lines 1240 to 1270 input the coefficients for equation ROW. Note the use of the fraction-getting subroutine in line 1250.

Exercise 17-2. In order to monitor the entering of the coefficients,

1. Insert line:

```
1265 PRINT "AT 1265, ROW = ";ROW;
" COL = ";COL;" AND COEF IS ";N(ROW,COL) ;"/";D(ROW,COL)
```

2. Insert line:

```
1275 END
```

3. Type RUN 1070, specify 3 variables and 2 equations, and enter the coefficients for equation (17-1). Note that ROW = 1 throughout, while COL goes from 1 to 3.

4. Remove line:

```
1275 END
```

LINES 1280 to 1290 get the constant for equation ROW.

Exercise 17-3. To monitor the entering of the constant term,

1. Insert line:

```
1295 PRINT "AT 1295, ROW = ";ROW;" VARS + 1 = ";VARS + 1;" AND
     CONSTANT = ";N(ROW, VARS + 1);"/";D(ROW, VARS + 1)
```

2. Insert line:

```
1296 END
```

3. Type RUN 1070, specify 3 variables and 2 equations, and enter the coefficients and constant for equation (17-1). Again note the values taken on by COL and ROW.
4. Remove line:

```
1296 END
```

We're about ready to observe the entering of all the equations. However, line 1310 calls for a subroutine to print equation ROW. You will write this subroutine in a minute. For now, we'll insert a pause and skip the printing.

Exercise 17-4.

1. Insert line:

```
1296 GET ANS$: GO TO 1360
```

2. Insert line:

```
1365 END
```

3. Type RUN 1070 and enter the coefficients and constants for both the equations (17-1) and (17-2). Note how ROW and COL change as the numbers are entered.
4. Delete lines 1265, 1295, and 1296.

Now it's your turn to write some of the program. We first need to be able to print the equations.

Exercise 17-5. Write a subroutine to print equation ROW. Start your subroutine with line number 1400. For COL = 1 to VARS, the subroutine must print N(ROW, COL), and "/", D(ROW, COL), and "X(" and COL and ")". Include "+" and "=" where appropriate. End with the constant N(ROW, VARS + 1);"/";D(ROW, VARS + 1).

Try out your subroutine by typing RUN 1070.

Exercise 17-6. Remove line:

```
1365 END
```

Exercise 17-7. Write a subroutine to print all the equations. Start your routine with line 2600. For ROW = 1 to EQNS, call your subroutine of Exercise 17-5 to print equation ROW. Try it out by typing RUN 1070 and entering equations (17-1) and (17-2).

Now we get to the core of the program—the operations on the equations.

Exercise 17-8. Write a subroutine to multiply an equation by a constant. Use lines with numbers larger than any you used for your printing subroutines. Following is an outline for the subroutine:

1. Ask the user which equation is to be multiplied.
2. Use INPUT ROW to record the answer.
3. Ask the user by what number equation ROW is to be multiplied.
4. Use the fraction-getter subroutine (GOSUB 820) to get the number as NUM/DEN.
5. Since the multiplication subroutine takes N1/D1 as one input, set N1/D1 to NUM/DEN.
6. For COL = 1 to VARS + 1, set N2/D2 = N(ROW,COL)/D(ROW,COL) and multiply by N1/D1, using the multiplication subroutine (GOSUB 470).
7. Put the answer N3/D3 in N(ROW,COL)/D(ROW,COL).
8. Print all the equations.
9. Return.

To try out your subroutine,

10. Type in a driver program:

```
10 GOSUB 1070:REM GET EQUATIONS
20 GOSUB ????:REM MULTIPLY AN EQUATION BY A CONSTANT
30 END
```

11. Type RUN, enter some equations, and try multiplying by constants.

Exercise 17-9. Write a subroutine to add one equation to another. The subroutine must add equation FST to equation SND and put the result in as the new equation SND.

1. Ask the user which equation (FST) is to be added to which (SND).
2. For COL = 1 to VARS + 1, add N(FST,COL)/D(FST,COL) to N(SND,COL)/D(SND,COL) and record the sum as N(SND,COL)/D(SND,COL).
3. Print all the equations.
4. Try out this subroutine by writing a driver program as you did for Exercise 17-8.

Finally, put these subroutines together to make a program to solve systems of equations in Exercise 17-10.

Exercise 17-10. Write a driver program that will get the system of equations from the user and present a menu of the options—add two equations, multiply an equation by a constant, or stop.

Exercise 17-11. Use your program to solve the following systems:

1. $$\begin{aligned} X_1 + 4X_2 &= -1 \\ 2X_1 + 3X_2 &= 3 \end{aligned}$$
2. $$\begin{aligned} 14X_1 + 5X_2 &= 7 \\ 7X_1 + 5X_2 &= 5 \end{aligned}$$
3. $$\begin{aligned} X_1 - 2X_2 + X_3 &= 3 \\ 2X_1 - 3X_2 + 4X_3 &= 1 \\ -3X_1 + X_2 + X_3 &= 5 \end{aligned}$$
4. $$\begin{aligned} -38X_1 + 48X_2 + 15X_3 &= -90 \\ 14X_1 - 21X_2 - 6X_3 &= 33 \\ 2X_1 - 3X_2 - X_3 &= 5 \end{aligned}$$

18

LINEAR EQUATIONS II

In Chapter 17, "Linear Equations I," we gave guidelines for writing a program to manipulate systems of linear equations. The program EQUATIONS is one such program. In this chapter, we use the program EQUATIONS to study the number of solutions a system can have.

Exercise 18-1. Use the program EQUATIONS to solve the system

$$\begin{aligned} (1) \quad 2X_1 - 3X_2 &= 2 \\ (2) \quad -4X_1 + X_2 &= 1 \end{aligned}$$

Write down your answer.

Exercise 18-2. Use EQUATIONS to solve the system of Exercise 18-1 again. Try to use a different sequence of operations this time. Do you get the same solution as before?

The system in Exercise 18-1 has a unique solution: $X_1 = -\frac{1}{2}$, $X_2 = -1$ is the only pair of numbers that satisfies both equations. Not all systems are this cooperative, however. Consider, for example, the system

$$\begin{aligned} (1) \quad 2X_1 - 4X_2 &= 3 \\ (2) \quad 2X_1 - 4X_2 &= 5 \end{aligned}$$

It is clear that there is no solution to this system: No matter what numbers you take for X_1

and X_2, 2 times the first minus 4 times the second cannot be both 3 and 5. We say that a system that has no solution is *inconsistent.*

Exercise 18-3. Use EQUATIONS on the inconsistent system mentioned above. Subtract one equation from the other and see what happens.

Some inconsistent equations are not quite so obvious as the one we've been discussing. For example, see the equations in Exercise 18-4.

Exercise 18-4. Use EQUATIONS to show that the following system is inconsistent:

$$\begin{aligned}
&(1)\ 2X_1 - X_2 + X_3 = 1\\
&(2)\ X_1 + X_2 + 4X_3 = 4\\
&(3)\ -X_1 + 2X_2 + 3X_3 = 1
\end{aligned}$$

You see from these examples of inconsistent systems that EQUATIONS eventually gives an equation of the form ZERO = ?, where ? is a nonzero constant. From such an equation, it is clear that the system is inconsistent.

There is one additional possibility for solutions to a system. Consider the following example:

$$\begin{aligned}
&(1)\ 3X_1 + 2X_2 = 5\\
&(2)\ 3X_1 + 2X_2 = 5
\end{aligned}$$

In this system, equation (2) is clearly redundant: It is exactly the same as equation (1). So, the system really reduces to just equation (1). However, the single equation $3X_1 + 2X_2 = 5$ has many solutions. To see this, rewrite the equation as $3X_1 = 5 - 2X_2$. Multiplying by $1/3$ results in

$$X_1 = \frac{5}{3} - \frac{2}{3} X_2$$

Now for any value you choose for X_2, this equation gives a value for X_1 that will result in a solution. For example, $X_2 = 0$ results in $X_1 = 5/3$. This is one solution. If you take different values for X_2, you get other solutions. A system that has more than one solution is called an *indeterminate system.*

Of course, indeterminate systems do not always reveal themselves as readily as the example above. However, when you attack an indeterminate system with EQUATIONS, you will eventually end up with more variables than equations.

Exercise 18-5. Use EQUATIONS to reduce the following system to just one equation with two variables, thus showing that the system is indeterminate. Also find at least two solutions for the system.

$$(1)\quad X_1 + \frac{1}{2} X_2 = \frac{1}{2}$$

$$(2)\quad 4X_1 + 2\,X_2 = 2$$

When an indeterminate system has more than two variables, you may have a wider choice of solutions. For example, consider the system

$$(1)\quad X_1 + X_3 = 1$$

$$(2)\quad X_2 - 2X_3 = 2$$

Solving these for X_1 and X_2 gives

$$(1)\quad X_1 = 1 - X_3$$

$$(2)\quad X_2 = 2 + 2X_3$$

so any choice you make for X_3 gives values for X_1 and X_2 to complete a solution. This situation is quite similar to our first indeterminate example: You have only one free-choice variable, X_3.

Now consider a different sort of indeterminate system:

$$(1)\quad X_1 + X_2 - X_3 = 2$$

Here we have only one equation but three variables. Solving for X_1 gives

$$(1)\quad X_1 = 2 - X_2 + X_3$$

Now you are free to choose both the variables X_2 and X_3 to be any numbers you want. Then you compute X_1 from this equation and you have a solution. Thus, you have two free-choice variables, X_2 and X_3.

Exercise 18-6. Use EQUATIONS to reduce the following system to just one equation with three values. Also find at least two solutions.

$$(1)\quad 60X_1 + 4X_2 - 3X_3 = 2$$

$$(2)\quad 15X_1 + X_2 - \frac{3}{4}X_3 = \frac{1}{2}$$

$$(3)\quad 20X_1 + \frac{4}{3}X_2 - X_3 = \frac{2}{3}$$

Exercise 18-7. Use EQUATIONS to show that the following system is indeterminate. Find at least two solutions.

$$\begin{aligned}(1)\ 5X_1 - 2X_2 + 7X_3 &= \frac{16}{5}\\ (2)\ 3X_1 - X_2 + 4X_3 &= \frac{1}{5}\\ (3)\ X_1 - \frac{1}{2}X_2 + \frac{3}{2}X_3 &= \frac{3}{2}\end{aligned}$$

In Exercises 18-8 to 18-17, use EQUATIONS to determine whether the system has a unique solution, is inconsistent, or indeterminate. If it is indeterminate, find at least two solutions.

Exercise 18-8.

$$\begin{aligned}(1)\ 3X_1 + 6X_2 - 6X_3 &= 4\\ (2)\ X_1 + 3X_2 - X_3 &= \frac{13}{6}\\ (3)\ X_1 + 2X_2 - X_3 &= \frac{5}{3}\end{aligned}$$

Exercise 18-9.

$$\begin{aligned}(1)\ 7X_1 + 10X_2 + 9X_3 &= 13\\ (2)\ X_1 + 2X_2 + 3X_3 &= 7\\ (3)\ 5X_1 + 6X_2 + 3X_3 &= 2\end{aligned}$$

Exercise 18-10.

$$\begin{aligned}(1)\ 10X_1 + 13X_2 + 14X_3 &= 43\\ (2)\ 6X_1 + 5X_2 - 2X_3 &= 15\\ (3)\ 5X_1 + 3X_2 - 6X_3 &= 8\end{aligned}$$

Exercise 18-11.

$$\begin{aligned}(1)\ 5X_1 - 2X_2 + 4X_3 &= -1\\ (2)\ 2X_1 - 5X_2 + 3X_3 &= -9\\ (3)\ X_1 - X_2 + X_3 &= -1\end{aligned}$$

Exercise 18-12.

(1) $4X_1 + 4X_2 - 5X_3 = 0$

(2) $8X_1 + 2X_2 - 10X_3 = -21$

(3) $2X_1 - 7X_2 + 5X_3 = -6$

Exercise 18-13.

(1) $16X_1 - X_2 + 12X_3 = 33$

(2) $14X_1 - 3X_2 + 13X_3 = 32$

(3) $20X_1 + 3X_2 + 10X_3 = 35$

Exercise 18-14.

(1) $-9X_1 + X_2 - 5X_3 - 8X_4 = -8$

(2) $21X_1 + 3X_2 + 10X_3 + 16X_4 = 8$

(3) $9X_1 + X_2 + 5X_3 + 8X_4 = 4$

(4) $3X_1 - X_2 + 5X_3 + 4X_4 = 5$

Exercise 18-15.

(1) $3X_1 - 4X_2 + 2X_3 + 4X_4 = -2$

(2) $X_1 + X_2 + 7X_3 + 7X_4 = 1$

(3) $X_2 + 3X_3 + 4X_4 = -1$

(4) $2X_1 - 3X_2 + X_3 + 5X_4 = -2$

Exercise 18-16.

(1) $X_1 - 2X_2 - 3X_3 - 8X_4 = -6$

(2) $X_1 - 7X_2 + 5X_3 = 1$

(3) $2X_1 - 3X_2 - X_3 - 5X_4 = -2$

(4) $6X_2 - 3X_3 + 3X_4 = 3$

Exercise 18-17.

$$\begin{aligned}
(1)\ -X_1 - 3X_2 + 18X_3 + 6X_4 &= 20\\
(2)\ X_1 - \frac{3}{2}X_2 + 12X_3 + 3X_4 &= \frac{25}{2}\\
(3)\ 3X_1 + 9X_2 - \frac{135}{2}X_3 - 20X_4 &= -\frac{137}{2}\\
(4)\ -X_1 - 3X_2 + 15X_3 + 5X_4 &= 17
\end{aligned}$$

During all this manipulating of equations, you may well have wondered whether the operations we have been performing always preserve the solution set. In other words, if you and a friend both start with the same system of equations and you each perform your own sequence of operations, will you end up with the same solutions? In fact, the solutions will be the same, provided there has been no round-off error in the computations (that's why we're using fractions) and you have used only legal operations. What operations are legal? Just the ones you have been using: Adding one equation to another and multiplying an equation by a constant. There is one exception: You certainly can't multiply an equation by zero. That would have the effect of removing that equation from the system.

We do not prove here that these operations preserve the solution set. Your previous work with EQUATIONS may already have convinced you that they do. As further evidence, the next exercise asks you to solve a system in three different ways. You will see that you get the same solution each time.

Exercise 18-18. For the system

$$\begin{aligned}
(1)\ 4X_1 + 5X_2 + 36X_3 &= 45\\
(2)\ 3X_1 + 4X_2 + 27X_3 &= 35\\
(3)\ 2X_1 + 2X_2 + 9X_3 &= 13
\end{aligned}$$

1. Reduce the system to the form

$$\begin{aligned}
(1)\ \quad X_2 \quad &= ?\\
(2)\ X_1 \quad\quad &= ?\\
(3)\ \quad\quad X_3 &= ?
\end{aligned}$$

To do this, first clear variable X_2 from equations (2) and (3) by attacking with equation (1). Then clear X_1 from equations (1) and (3) by attacking with equation (2). Finally, clear X_3 from equations (1) and (2) by attacking with equation (3). Write down your solution.

2. Reduce the system to the form

$$\begin{array}{llll} (1) & & & X_3 = ? \\ (2) & & X_2 & = ? \\ (3) & X_1 & & = ? \end{array}$$

3. Reduce the system to the form

$$\begin{array}{llll} (1) & & X_2 & = ? \\ (2) & & & X_3 = ? \\ (3) & X_1 & & = ? \end{array}$$

Are the solutions always the same?

In addition to solving a given system of equations, you can use the program EQUATIONS to create systems with a prescribed solution. The next two exercises illustrate this.

Exercise 18-19. Find a system of equations, with no coefficients equal to zero, whose solution is $X_1 = 1$, $X_2 = 2$, and $X_3 = 3$.

Exercise 18-20. Find a system of equations with all coefficients as nonzero integers whose solution is $X_1 = 1/2$, $X_2 = 1/3$, and $X_3 = 1/5$.

19

LINEAR EQUATIONS III

In the preceding chapter, you used the computer to assist in solving systems of linear equations. With the program EQUATIONS, all you had to do was tell the computer which two equations to add or which equation to multiply by a constant, and you eventually got the solution to the system. Now let's make the computer take over even more of the work by developing an algorithm for solving a system automatically. When you have such a program, all you'll have to do is enter the equations, tell the computer to get busy, and wait for the solution.

First we need some terminology, so let's look at another system.

Exercise 19-1. Use EQUATIONS to solve the following system. Follow steps 1 to 3 given below.

$$(1)\ 4X_1 + 2X_2 + 3X_3 = 4$$

$$(2)\ 3X_1 - X_2 + X_3 = 7$$

$$(3)\ 2X_1 - 5X_2 + 7X_3 = 9$$

1. Use equation (1) to clear variable X_1 from equations (2) and (3), as follows:
   - a. To eliminate X_1 from equation (2), get a coefficient of -3 on X_1 in equation (1), then add equation (1) to equation (2).
   - b. To eliminate X_1 from equation (3), get a coefficient of -2 on X_1 in equation (1), then add equation (1) to equation (3). During this process of using equation (1) to clear variable X_1, equation (1) is called the *pivot equation* and X_1, the *pivot variable*.

c. to simplify equation (1) with respect to X_1, reduce the coefficient of X_1 in that equation to a value of 1.

Note how you used additive and multiplicative inverses of fractions in this clearing process: To get a variable to cancel, you must use the additive inverse of its coefficient. To get a coefficient of 1 on a variable, you must use the multiplicative inverse of its coefficient.

The result of this first clearing should be

$$\begin{aligned} &(1)\quad X_1 + \frac{1}{2}X_2 + \frac{3}{4}X_3 = 1 \\ &(2)\quad -\frac{5}{2}X_2 - \frac{5}{4}X_3 = 4 \\ &(3)\quad -6X_2 + \frac{11}{2}X_3 = 7 \end{aligned}$$

2. Now use equation (2) to clear variable X_2 from equations (1) and (3). During this clearing, equation (2) is the pivot equation and X_2 the pivot variable. Leave X_2 with coefficient 1 in equation (2). The result should be

$$\begin{aligned} &(1)\quad X_1 + \frac{1}{2}X_3 = \frac{9}{5} \\ &(2)\quad X_2 + \frac{1}{2}X_3 = \frac{-8}{5} \\ &(3)\quad \frac{17}{2}X_3 = \frac{-13}{5} \end{aligned}$$

3. Finally, use equation (3) as the pivot equation to clear variable X_3 from equations (1) and (2). The result should be as follows:

$$\begin{aligned} &(1)\quad X_1 = \frac{166}{85} \\ &(2)\quad X_2 = \frac{-123}{85} \\ &(3)\quad X_3 = \frac{-26}{85} \end{aligned}$$

So now we have the solution.

There are two main parts of the procedure we used for solving this system. We (a) chose a pivot equation and pivot variable, and, (b) used the pivot equation and pivot variable to clear that variable from the other equations. We repeated parts a and b until we had the solution to the system.

The program EQUATIONS CLEAR does the variable clearing of part b, once you specify the pivot equation and pivot variable. We'll work with this program now to see how to make the computer do the equation and variable choosing of part a. The program contains the same subroutines for adding and multiplying equations as does EQUATIONS.

Listing 19-1 shows the subroutine in EQUATIONS CLEAR that clears a variable.

LISTING 19-1

```
2570  REM   SUBROUTINE TO CLEAR A VARIABLE
2580  HOME
2590  PRINT : PRINT "CLEAR WHICH VARIABLE";
2600  INPUT PVAR$:PVAR =  VAL ( RIGHT$ (PVAR$,1))
2610  PRINT "USING WHICH EQUATION";
2620  INPUT PEQN
2630  REM   REDUCE PIVOT COEFFICIENT TO 1
2640  REM  SET UP FOR MULT SBRTN
2650 ROW = PEQN
2660 N1 = D(PEQN,PVAR):D1 = N(PEQN,PVAR): REM   NOTE RECIPROCA
L
2670  GOSUB 1780: REM    MULT EQN "ROW" BY N1/D1
2680  FOR RW = 1 TO EQNS
2690  REM  SKIP PIVOT EQN AND THOSE WITH COEF ON X(PVAR) ALREA
DY 0:
2700  IF RW = PEQN OR N(RW,PVAR) = 0 THEN 2880
2710  REM  MULT PEQN SO AS TO CANCEL
2720  REM  MULT PIVOT EQN BY ADDITIVE INVERSE
2730  REM   OF COEF OF X(PVAR) IN EQN RW:
2740  REM   SET UP FOR MULTIPLYING SUBROUTINE:
2750 ROW = PEQN
2760 N1 =  - N(RW,PVAR):D1 = D(RW,PVAR): REM  NOTE ADDITIVE
INVERSE
2770  GOSUB 1780: REM    MULT EQN "ROW" BY N1/D1
2780  REM   ADD NEW EQUATION PEQN TO EQN RW
2790  REM   TO CANCEL X(PVAR) TERM FROM EQN RW:
2800 FST = PEQN:SND = RW: GOSUB 1960
2810  PRINT
2820  REM   RESTORE PIVOT EQUATION
2830  IF N(PEQN,PVAR) = 1 AND D(PEQN,PVAR) = 1 THEN 2880: REM
 SKIP RESTORING
2840  REM  SET UP FOR MULT SUBRTN:
2850 ROW = PEQN
2860 N1 = D(PEQN,PVAR):D1 = N(PEQN,PVAR): REM  NOTE RECIPROCAL

2870  GOSUB 1780: REM  MULT SUBRTN
2880  NEXT : REM   CLEAR X(PVAR) FROM NEXT EQN
2890  RETURN
```

Let's look at the subroutine to see how it works.

Lines 2590 to 2620 establish the pivot variable as variable X_{PVAR} and the pivot equation as equation number PEQN. Line 2600 obtains PVAR$ as a string and then extracts the right-most character of the string for PVAR. This allows you to enter the variable either as XK or as K.

Lines 2630 to 2670 reduce the coefficient on X_{PVAR} in the pivot equation to 1 by multiplying by the reciprocal of the coefficient of X_{PVAR}. Lines 2650 and 2660 set up the variables ROW, N1, and D1, for the subroutine that multiplies equation ROW by a constant N1/D1. Line 2670 calls this equation multiplying subroutine.

Lines 2680 and 2880 are the beginning and end of a FOR-NEXT loop that clears the variable X_{PVAR} from the nonpivot equations. The loop uses the variable RW to designate a nonpivot equation.

Lines 2690 to 2700 skip the variable clearing if equation RW is the pivot equation, or if the pivot variable already has coefficient 0 in equation RW.

If the coefficient of the pivot variable in equation RW is not zero, we must get its additive inverse as the coefficient in the pivot equation. Since we earlier reduced the coefficient in the pivot equation to 1, we need only multiply by this additive inverse. Lines 2750 to 2760 set up the variables ROW, N1, and D1 for the subroutine that multiplies equation ROW by N1/D1. Line 2760 makes N1/D1 equal to the additive inverse.

Lines 2780 to 2800 add the pivot equation to equation RW to achieve the cancellation of the pivot variable from equation RW.

Lines 2820 to 2870 restore the coeffocient of the pivot variable in the pivot equation to 1.

To get some experience with EQUATIONS CLEAR, use it to solve the systems of Exercises 19-2 and 19-3.

Exercise 19-2.

$$\begin{aligned}
&(1)\quad X_1 + X_2 - X_3 = 0\\
&(2)\quad 2X_1 - X_2 + 3X_3 = 1\\
&(3)\quad 5X_1 + 2X_2 - X_3 = -1
\end{aligned}$$

Exercise 19-3.

$$\begin{aligned}
&(1)\quad X_2 + 2X_3 = \frac{-1}{2}\\
&(2)\quad X_1 - 7X_2 - X_3 = 5\\
&(3)\quad 2X_1 + \frac{1}{2}X_2 - X_3 = 1
\end{aligned}$$

In order to have the computer solve a system automatically, we must be able to tell it how to choose the pivot equations and variables. The next two exercises show that there are various sequences of choices that will work.

Exercise 19-4. Follow steps 1 to 3 to solve this system:

$$\begin{array}{lrcl}
(1) & 3X_2 - X_3 & = & 3 \\
(2) & 3X_1 + 9X_2 - 2X_3 & = & 15 \\
(3) & 2X_1 + 12X_2 - 3X_3 & = & 17
\end{array}$$

1. Use equation (3) to clear variable X_1. The result is

$$\begin{array}{lrcl}
(1) & 3X_2 - X_3 & = & 3 \\
(2) & -9X_2 + \frac{5}{2}X_3 & = & \frac{-21}{2} \\
(3) & X_1 + 6X_2 - \frac{3}{2}X_3 & = & \frac{17}{2}
\end{array}$$

2. Now use equation (1) to clear variable X_2. The result is

$$\begin{array}{lrcl}
(1) & X_2 - \frac{1}{3}X_3 & = & 1 \\
(2) & -\frac{1}{2}X_3 & = & \frac{-3}{2} \\
(3) & X_1 + \frac{1}{2}X_3 & = & \frac{5}{2}
\end{array}$$

3. Finally, use equation (2) to clear variable X_3. The result is

$$\begin{array}{lrcl}
(1) & X_2 & = & 2 \\
(2) & X_3 & = & 3 \\
(3) & X_1 & = & 1
\end{array}$$

Exercise 19-5. Solve the system of Exercise 19-4 again.

1. Clear X_1 using equation (2). The result is

$$\begin{array}{lrcl}
(1) & 3X_2 - X_3 & = & 3 \\
(2) & X_1 + 3X_2 - \frac{2}{3}X_3 & = & 5 \\
(3) & 6X_2 - \frac{5}{3}X_3 & = & 7
\end{array}$$

2. Clear X_3 using equation (1). The result is

$$\begin{array}{lrcl}
(1) & -3X_2 + X_3 & = & -3
\end{array}$$

$$\begin{array}{lrcl}
(2) & X_1 + X_2 & = & 3 \\
(3) & X_2 & = & 2
\end{array}$$

3. Clear X_2 using equation (3).

To see that you cannot choose the pivot equations and variables with wild abandon, look at what happens in Exercise 19-6.

Exercise 19-6. Using the system of Exercise 19-4, do the following:

1. Clear X_1 using equation (2). The result is

$$\begin{array}{lrcl}
(1) & 3X_2 - X_3 & = & 3 \\
(2) & X_1 + 3X_2 - \frac{2}{3}X_3 & = & 5 \\
(3) & 6X_2 - \frac{5}{3}X_3 & = & 7
\end{array}$$

2. Clear X_2 using equation (2). The result is

$$\begin{array}{lrcl}
(1) & -X_1 - \frac{1}{3}X_3 & = & -2 \\
(2) & \frac{1}{3}X_1 + X_2 - \frac{2}{9}X_3 & = & \frac{5}{3} \\
(3) & -2X_1 - \frac{1}{3}X_3 & = & -3
\end{array}$$

Here, in step 1, you cleared X_1 using equation (2). Then, in step 2, you used equation (2) again and put the X_1 variables right back in! Thus, reusing a previous pivot equation destroys the earlier clearing. Therefore, each new pivot equation must be chosen to be different from all previous pivot equations. Likewise, each new pivot variable must be chosen to be different from all previous pivot variables.

To make this choosing easy enough for the stupid computer to do on its own, work from the top down and from left to right through the system. First, use equation (1) to clear variable X_1, then equation (2) to clear variable X_2, and so forth. But what if equation (1) doesn't have an X_1 term? For example, this occurs in the system of Exercise 19-4:

$$\begin{array}{lrcl}
(1) & 3X_2 - X_3 & = & 3 \\
(2) & 3X_1 + 9X_2 - 2X_3 & = & 15 \\
(3) & 2X_1 + 12X_2 - 3X_3 & = & 17
\end{array}$$

In this system, equation (1) can't be used to clear variable X_1 because it has no X_1 term. You can, however, interchange equation (1) with an equation that does have an X_1 variable, for instance, equation (2). The result is

$$\begin{array}{lrcrcrcr}
(1) & 3X_1 & + & 9X_2 & - & 2X_3 & = & 15 \\
(2) & & & 3X_2 & - & X_3 & = & 3 \\
(3) & 2X_1 & + & 12X_2 & - & 3X_3 & = & 17
\end{array}$$

Now equation (1) can be used to clear variable X_1. This necessity for interchanging equations is why EQUATIONS CLEAR has an interchange option.

The equation interchanging subroutine of EQUATIONS CLEAR is given in Listing 19-2.

LISTING 19-2

```
2150  REM  SUBRTN TO INTERCHANGE EQNS
2160  REM     SWITCH EQNS "FST" AND "SND"
2170  HOME
2180  PRINT : PRINT "INTERCHANGE WHICH EQUATIONS"
2190  PRINT : PRINT "FIRST=";: INPUT FST
2200  PRINT "SECOND=";: INPUT SND
2210  FOR COL = 1 TO VARS + 1
2220  REM   HOLD COEF OF X(COL) FROM EQN FST:
2230 NT = N(FST,COL):DT = D(FST,COL)
2240  REM  NOW SWITCH:
2250 N(FST,COL) = N(SND,COL)
2260 D(FST,COL) = D(SND,COL)
2270 N(SND,COL) = NT
2280 D(SND,COL) = DT
2290  NEXT : GOSUB 2060: REM   PRINT EQUATIONS
2300  RETURN
```

Lines 2180 to 2200 obtain the equations FST and SND to be interchanged.

Lines 2210 to 2290 are for a FOR-NEXT loop that interchanges the coefficient of X_{COL} in equation FST with its coefficient in equation SND. During the switching process, the coefficient from equation FST is held temporarily as NT/DT.

Now use the interchange option of EQUATIONS CLEAR to solve some systems in the rigid fashion the computer will use.

Exercise 19-7. Solve the system of Exercise 19-4. Always work from top to bottom and left to right, interchanging equations when necessary.

Exercise 19-8. Solve the system

$$\begin{array}{lrcrcrcr}
(1) & & & 2X_2 & + & 3X_3 & = & 4 \\
(2) & & & & - & 4X_3 & = & 7 \\
(3) & 5X_1 & + & X_2 & - & X_3 & = & 10
\end{array}$$

You must clear X_1 using an equation (1), then clear X_2 using an equation (2), and then clear X_3 using an equation (3).

Exercise 19-9. Solve the system

$$\begin{array}{lrrrrl}
(1) & 4X_1 + & 2X_2 - & X_3 + & X_4 & = 5 \\
(2) & 2X_1 + & X_2 + & 4X_3 - & 6X_4 & = 7 \\
(3) & & & 2X_3 - & X_4 & = 7 \\
(4) & & & X_3 + & 5X_4 & = 9
\end{array}$$

In this system, when you clear X_1 using equation (1), X_2 disappears from the system, so you must move on to X_3 as a pivot variable.

Now that you see how to specify the selecting of the pivot equation and variables, you can make the computer do the whole solving job.

Exercise 19-10. Modify EQUATIONS CLEAR to make it solve a system automatically. Add a driver program, starting with line 4000, which chooses the pivot equations and variables and uses the interchanging and clearing subroutines.

When you need to interchange two equations, set up the variables FST and SND and call the interchanging subroutine by GOSUB 2210.

When you have set the pivot variable with PVAR and the pivot equation with PEQN, and are ready to clear the variable PVAR, call the clearing subroutine by GOSUB 2630.

The equation-getting subroutine starts at line 850, so use GOSUB 850 to set up the equations. To get your program started, insert a line

```
10 GOTO 4000
```

Figure 19-1 is a flow chart for this driver program. The flow chart uses a variable RW to designate an equation while the program searches for an equation that has a nonzero coefficient on the pivot variable. When your program is working, try it out on the systems in Chapter 18, "Linear Equations II."

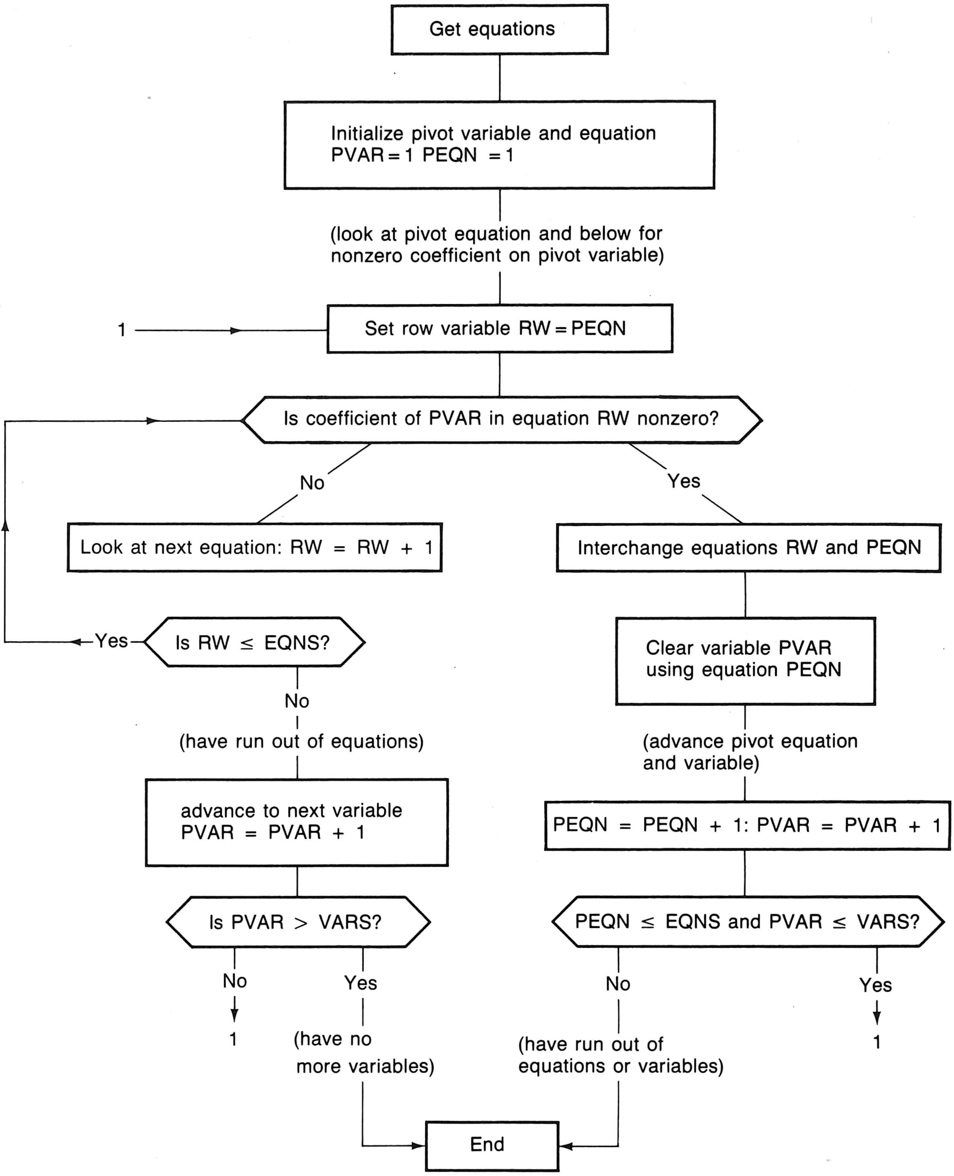

Get equations
Initialize pivot variable and equation
PVAR = 1 PEQN = 1
(look at pivot equation and below for nonzero coefficient on pivot variable)
1
Set row variable RW = PEQN
Is coefficient of PVAR in equation RW nonzero?
No
Yes
Look at next equation: RW = RW + 1
Interchange equations RW and PEQN
Yes
Is RW ≤ EQNS?
No
(have run out of equations)
Clear variable PVAR using equation PEQN
(advance pivot equation and variable)
advance to next variable
PVAR = PVAR + 1
PEQN = PEQN + 1: PVAR = PVAR + 1
Is PVAR > VARS?
PEQN ≤ EQNS and PVAR ≤ VARS?
No
1
Yes
(have no more variables)
No
(have run out of equations or variables)
Yes
1
End

Figure 19-1 Flow chart for solving system of equations

20

LINEAR EQUATIONS AND ROUND-OFF ERROR

Whenever you use a computer to perform computations with numbers, round-off error may affect the accuracy of your results. This is because the computer, being a finite machine, must round off any decimal that has more places than the machine can carry. We discussed some of the effects of round-off error in Chapter 1. We saw there that the computer not only makes errors, but it also tries to cover them up.

In this chapter, we investigate the sometimes dramatic effect of round-off error in solving systems of linear equations. To keep things simple, we will work only with equations in two variables X and Y. The program XY EQUATIONS allows you to manipulate the equations in the same manner as you did with the program EQUATIONS in Chapter 17, "Linear Equations I." In fact, XY EQUATIONS is essentially the same program as EQUATIONS except that the number of variables is restricted to 2 and the variables are printed as X and Y rather than as X_1 and X_2. So, with XY EQUATIONS, you can add two equations and multiply an equation by a constant.

As a warm-up with XY EQUATIONS, solve the systems in Exercises 20-1 and 20-2.

Exercise 20-1.

$$(1)\quad X + 2Y = \frac{1}{3}$$

$$(2)\quad 3X - 4Y = \frac{1}{7}$$

Exercise 20-2.

$$\begin{aligned}(1)\ 7X - Y &= 123\\ (2)\ X + 5Y &= 117\end{aligned}$$

The program XY EQUATIONS works with fractions, so there is no round-off error involved (assuming you do not use any numerator or denominator with too many digits for the computer to handle). Hence, you should get the same solution for a system no matter what sequence of operations you use. Let's check this out before getting into the round-off error.

Exercise 20-3. Use XY EQUATIONS to solve the system

$$\begin{aligned}(1)\ 14X + 13Y &= 54\\ (2)\ 11X + 10Y &= -23\end{aligned}$$

in the following four ways. Write down your solution each time and check that you get the same solution all four times.

1. Reduce the system to the form

$$\begin{aligned}(1)\ X \qquad &= ?\\ (2)\qquad Y &= ?\end{aligned}$$

by first eliminating X from equation (2) and then eliminating Y from equation (1).

2. Reduce the system to the form of step 1, but this time first eliminate Y from equation (1), then eliminate X from equation (2).
3. Reduce the system to the form

$$\begin{aligned}(1)\qquad Y &= ?\\ (2)\ X \qquad &= ?\end{aligned}$$

by first eliminating X from equation (1), then eliminating Y from equation (2).

4. Finally, reduce to the form of step 3 again, this time by first eliminating Y from equation (2), then X from equation (1).

You see that you get the same solution by each of the four prescribed methods of Exercise 20-3. This should make it seem reasonable that you will get the same solution for *any* sequence of legal operations. (The only illegal operation allowed by the program XY EQUATIONS is multiplying an equation by 0.)

Now we'd like to introduce the rounding-off of numbers into the arithmetic of the program XY EQUATIONS. We could replace the program's fraction arithmetic with the computer's own built-in decimal arithmetic, but since the computer carries some eight decimal places, this would make the effect of the error difficult to see. Instead, we will round off all numbers at two decimal places. To do this, we need a formula for rounding, so let's see how to make the computer round off a number.

Rounding-off can be done as follows. Suppose we want to round off 3.14159 to two decimal places. Do the following steps:

1. Multiply by 10^2; the result is 314.159.
2. Add 0.5; the result is 314.659.
3. Take integer part; the result is 314.
4. Divide by 10^2; the result is 3.14.

So we got 3.14, just what you expected for rounding-off 3.14159 to two places.

Look at another example: Round off 23.4265 to two places. Here you expect the result to be 23.43. The rounding-off can be accomplished by the same sequence of operations as in the first example:

1. Multiply by 10^2; the result is 2,342.65.
2. Add 0.5; the result is 2,343.15.
3. Take integer part; the result is 2,343.
4. Divide by 10^2; the result is 23.43.

To find a general formula for rounding-off a number X, follow the same steps for X:

1. Multiply by 10^2; the result is $X*10^2$.
2. Add 0.5; the result is $X*10^2 + 0.5$.
3. Take integer part; the result is $\mathrm{INT}(X*10^2 + 0.5)$.
4. Divide by 10^2; the result is

$$\mathrm{INT}(X*10^2 + 0.5)/10^2 \tag{20-1}$$

Equation (20-1) is the general formula for rounding-off to two places. If you want to round off to some other number E of places, replace the exponent 2 in formula (20-1) by E.

The following program uses formula (20-1) in line 110 to define a round-off function RO.

```
100 E = 2: REM ROUND TO E PLACES
110 DEF FN RO (X) = INT (X*10^E+.5) /10^E
120 PRINT "WHAT NUMBER DO YOU WANT TO ROUND";
130 INPUT A
140 PRINT A; "ROUNDED TO ";E;" PLACES IS ";FN RO (A)
```

Exercise 20-4. Type in and run this program for various numbers to check that the formula in line 110 really works. Also change E in line 100 and try rounding off to a different number of places.

Now let's modify XY EQUATIONS so that, instead of doing fraction arithmetic, it does two-place decimal arithmetic. To do this, we'll convert each fraction to a decimal quotient and then round off the decimal to two places. We must do this when numbers are entered at the start and each time a computation is performed. The following exercise gives the necessary modifications to XY EQUATIONS.

Exercise 20-5. Modify XY EQUATIONS as follows:

1. To set up the rounding-off function, insert lines

```
10 E = 2: REM ROUND TO E PLACES
20 DEF FN RO(X) = INT(X*10^E+.5)/10^E
```

2. Now modify the fraction-getting subroutine, lines 830 to 1060. Line 990 returns numerator NUM and denominator DEN when there is no "/" in the entered string. To round off NUM, insert line

```
985 NUM = FN RO(NUM)
```

   Line 1060 returns numerator NUM and denominator DEN. To convert to a rounded decimal, insert line

```
1055 NUM = FN RO(NUM/DEN) : DEN = 1
```

3. Modify the fraction-multiplying subroutine, lines 480 to 570 as follows: Line 570 returns the product as a fraction having numerator NNUM and denominator NDEN after having reduced the fraction product in line 540. To convert to a reduced decimal and skip the reducing, insert lines

```
535 NNUM = FN RO(NUM/DEN) : NDEN=1
536 GO TO 560: REM SKIP REDUCING
```

4. Modify the fraction-adding subroutine, lines 720 to 810, as follows: Line 810 returns the sum as a fraction with numerator NNUM and denominator NDEN. To convert to a rounded decimal and skip the reducing, insert lines

```
775 NNUM = FN RO(NUM/DEN) : NDEN=1
776 GO TO 800: REM SKIP REDUCING
```

With the modifications noted above, XY EQUATIONS computes strictly with two-place decimals. Note that numbers may still be entered in the form N/D, but the program converts any fraction to a two-place decimal before starting any computations with it.

Because the program uses rounded decimals, it is more difficult to find multiplicative inverses: It may take several multiplications by reciprocals to reduce a coefficient to 1.

Exercise 20-6. Use this rounding form of XY EQUATIONS to solve the system of Exercise 20-3 in each of the four ways specified in that exercise. Compare your answers with the correct fraction answers and with one another. Are the results surprising?

For an even more dramatic illustration of the effects of round-off error, try the next exercise.

Exercise 20-7. Use the rounding version of XY EQUATIONS to solve the system

$$\begin{aligned}(1)\ 145X + 136Y &= 32\\ (2)\ 111X + 104Y &= 77\end{aligned}$$

in the four ways listed in Exercise 20-3. Record and compare your answers. Does this system of equations have a unique solution? How can you possibly tell from the results of this program?

These last two exercises show that round-off error can have a dramatic impact on solving systems of linear equations. Depending on the sequence of operations you use, you may get wildly different results. You may think that since we've rounded to two decimal places, we have created the bizarre results. Well, every computer rounds off decimals at some number of places, so round-off error will *always* be present. Of course, if the computer carries several decimal places in its computations, you may not notice the effect of the rounding-off. But what if you are solving a large system, for instance, 100 equations in 100 variables. With the thousands of computations involved, how can you have any confidence in the computer's results?

To know how accurate the results of a program are, you must be able to estimate the round-off error. To get results that are as accurate as possible, you must be able to construct good algorithms. We cannot investigate these problems in detail here, but if you want more information, look into the area of mathematics called *numerical analysis*.

21

LINES

The ability of the computer to draw pictures on its screen provides you with a powerful tool for investigating certain areas of mathematics. You can draw lines and graph functions. You can also model motion by making a point move around on the screen. In this chapter, we introduce some of the graphics commands for the APPLE and use them in a program that graphs straight lines.

We need to use the following commands.

1. HGR sets the computer to high-resolution graphics mode.
2. HCOLOR = X sets the color for plotting.
3. HPLOT U,V lights a spot on the screen at the location with coordinates U, V.
4. TEXT returns the computer to the usual text screen.

Now try out these commands in Exercise 21-1.

Exercise 21-1.

1. Type VTAB 23 to make the text appear below the graphics screen.
2. Type HGR to switch to high-resolution graphics.
3. Type HCOLOR = 3 to set the color.
4. Type HPLOT U,V for various U and V.
5. Type TEXT to return to text mode.

If you have a color monitor, you may want to experiment with other HCOLOR settings. However, for other HCOLOR values, HPLOT lights the spot for certain values of U and V and not for others. Hence, the setting HCOLOR = 3 is most convenient for our graphics work.

In the command HPLOT U,V, the variable U runs from U = O at the left-hand edge of the screen to U = 279 at the right-hand edge. V runs from V = O at the top of the screen to V = 159 just above the text lines.

The next exercise gives you some more experience with the HPLOT command.

Exercise 21-2. Test out the HPLOT command by typing in and running the following program:

```
100 HGR
110 HCOLOR = 3
120 FOR K = 1 TO 100
130 U = 2*K
140 V = K
150 HPLOT U, V
160 NEXT K
```

To return to the text screen, type TEXT. Also try this program with some different arrangements of the coordinates, such as U = 2*K and V = 3*K.

Since we are accustomed to using the coordinates X and Y to describe points in the plane, we'd like to use them for our graphing here too. Unfortunately, the variables U and V in the command HPLOT U,V do not correspond very well to our X and Y: U and V are restricted to certain positive values, and V increases as the point moves down from the top of the screen. Hence we must find out how to convert our X and Y coordinates for the plane into U and V coordinates for the computer screen.

To plot a portion of the X-Y plane on the U-V screen, we must convert X coordinates into U coordinates and Y into V. The easiest conversion is a linear (straight-line) assignment, so here is a chance to use your knowledge of the equations of straight lines. Suppose the range of X values we want to use runs from X = XL to X = XH. (XL stands for "low X" and XH for "high X.") We must convert these X values into U values, with X = XL going to U = 0 and X = XH going to U = 279. So we have two X-U points (XL, 0) and (XH, 279) on the X-U line. The slope of this X-U line is (279 − 0)/(XH − XL). (XL, 0) is a point on the line, so the point-slope form for the equation of the line is

$$U - 0 = \frac{279 - 0}{XH - XL}(X - XL)$$

Rewriting this gives

$$U = 279*(X - XL)/(XH - XL) \tag{21-1}$$

This function converts the X values in the interval

$$XL \leq X \leq XH$$

to U values in the interval

$$0 \leq U \leq 279$$

Now let's find how to convert Y values to V values. If the range of Y values we want to use is $YL \leq Y \leq YH$, we must convert this interval to the V interval $0 \leq V \leq 159$. Since V increases from the top down and Y from the bottom up, we want $Y = YL$ to go to $V = 159$ and $Y = YH$ to go to $V = 0$. Thus, we have two Y-V points (YL, 159) and (YH, 0) on the Y-V line. The slope of this line is

$$\frac{0 - 159}{YH - YL} = \frac{-159}{YH - YL}$$

We have the point (YH, 0) on the line, so the point-slope form of the equation is

$$V - 0 = \frac{-159}{YH - YL}(Y - YH)$$

Rewriting this gives

$$V = -159*(Y - YH)/(YH - YL) \qquad (21\text{-}2)$$

This function converts the Y values in the interval

$$YL \leq Y \leq YH$$

to V values in the interval

$$0 \leq V \leq 159$$

The plotter subroutine in Listing 21-1 uses formulas (21-1) and (21-2) to convert X-Y values to U-V values.

LISTING 21-1

```
120  REM  PLOTTER SUBROUTINE
130  REM
140  REM  CONVERT X,Y TO U,V:
150  REM
160  REM   X=XL TO U=0
170  REM   X=XH TO U=279
180  REM
190  REM   Y=YL TO V=159
200  REM  Y=YH TO V=0
210  REM
220 U = 279 * (X - XL) / (XH - XL)
230 V =  - 159 * (Y - YH) / (YH - YL)
240  REM     IF U OR V OUT OF RANGE, RETURN:
250  IF U < 0 OR U > 279 THEN 280
260  IF V < 0 OR V > 159 THEN 280
270  HPLOT U,V
280  RETURN
```

Lines 220 and 230 convert X and Y to U and V by the formulas given above.

Lines 250 and 260 check whether U or V is out of the allowed range for the HPLOT command. If either is out of range, the subroutine returns control to the main program without attempting to plot.

If both U and V are in the allowed range, line 270 plots the point.

This subroutine will be a great help to you in doing graphics programming. By having it available, you can do your thinking and programming in X and Y and just call the subroutine to do the converting and plotting.

Now that you see how plotting of points in the X-Y plane can be done, let's graph some straight lines. The program LINES (see Listing 21-2) contains the plotter subroutine we just discussed.

LISTING 21-2

```
100  REM    LINES
110  GOTO 290
120  REM  PLOTTER SUBROUTINE
130  REM
140  REM  CONVERT X,Y TO U,V:
150  REM
160  REM   X=XL TO U=0
170  REM   X=XH TO U=279
180  REM
190  REM   Y=YL TO V=159
200  REM  Y=YH TO V=0
210  REM
220 U = 279 * (X - XL) / (XH - XL)
230 V =  - 159 * (Y - YH) / (YH - YL)
240  REM    IF U OR V OUT OF RANGE, RETURN:
250  IF U < 0 OR U > 279 THEN 280
260  IF V < 0 OR V > 159 THEN 280
270  HPLOT U,V
280  RETURN
290  REM  MAIN PROGRAM
300  HOME
310  PRINT "      THIS PROGRAM PLOTS LINES "
320  PRINT : PRINT "Y = MX+B."
330  PRINT : PRINT "YOU SUPPLY THE SLOPE M"
340  PRINT : PRINT "AND Y-INTERCEPT B."
350  VTAB 23: PRINT "PRESS ANY KEY TO CONTINUE."
360  GET ANS$
370  HOME
380  REM  SET HIGH AND LOW X AND Y BOUNDS:
390 XL =  - 10
400 XH = 10
410 YL =  - 10
420 YH = 10
430  REM  SET X AND Y STEP SIZES:
440 XS = (XH - XL) / 140: REM  140 STEPS XL TO XH

450 YS = (YH - YL) / 159: REM  159 STEPS YL TO YH
460  HGR : HCOLOR= 3
470  POKE 34,20
480  HOME
490  PRINT "PLOTTING X AXIS"
500  REM    PLOT X AXIS:
510 Y = 0
520  FOR X = XL TO XH STEP XS
530  GOSUB 120: REM   PLOTTER SUBROUTINE
540  NEXT
```

```
550  HOME
560  PRINT "PLOTTING Y AXIS"
570  REM  PLOT Y AXIS:
580 X = 0
590  FOR Y = YL TO YH STEP YS
600  GOSUB 120: REM   PLOTTER SUBROUTINE
610  NEXT
620  HOME
630  REM   NOW PLOT LINE Y = MX+B:
640  PRINT " TO GRAPH Y = MX + B "
650  PRINT : PRINT "WHAT SLOPE M DO YOU WANT";
660  INPUT M
670  PRINT : PRINT "WHAT INTERCEPT B DO YOU WANT";
680  INPUT B
690  HOME
700  PRINT "PLOTTING Y = ";M;"X + ";B
710  FOR X = XL TO XH STEP XS
720 Y = M * X + B
730  GOSUB 120: REM   PLOTTER SUBROUTINE
740  NEXT
750  PRINT : PRINT "PLOT ANOTHER LINE (Y OR N)";
760  INPUT ANS$
770  IF ANS$ <  > "N" THEN 620: REM    GET ANOTHER M,B
780  HGR : TEXT : HOME : END
```

This program draws the graphs of straight lines using the slope-intercept form Y = MX + B. You specify the slope M and intercept B. Now let's analyze the LINES program.

Lines 120 to 280 contain the plotter subroutine, which we have already discussed.

Lines 370 to 420 set the X and Y intervals to $-10 \le X \le 10$ and $-10 \le Y \le 10$, respectively.

Line 440 establishes the step size XS to be used in plotting the lines. We will plot 140 points with equally spaced X coordinates, so line 440 computes the distance between the X coordinates as (XH − XL)/140. You could plot as many as 280 X coordinates, because U goes from 0 to 279, but that takes twice as long as 140 points, and 140 points seem to give good enough lines.

Line 450 computes the Y step size in a similar fashion. We need this to plot the Y axis.

Lines 460 to 480 switch to the graphics screen, protect the text lines from scrolling, and clear the text portion of the screen.

Lines 490 to 540 draw the X axis. First, Y is set to 0. Then X is started off at XL and incremented by the step size XS until X = XH. For each step, the plotter subroutine is called to plot the point.

Lines 550 to 610 plot the Y axis in a similar fashion.

Lines 620 to 680 obtain the slope M and intercept B.

Lines 690 to 740 draw the line Y = MX + B. The FOR-NEXT loop increments X by the step size XS and calls the plotter subroutine.

Exercise 21-3. To see how LINES works, use it to draw several lines.

Exercise 21-4. Draw several lines with the Y intercept 0. Use both positive and negative slopes. Check out the effect of the slope: Positive slope lines rise as you move to the right, and larger slopes give a faster rise. Negative slope lines fall as you move to the right.

Exercise 21-5. Draw the graphs of

1. $Y = X + 2$
2. $Y = 2X + 4$
3. $Y = 4X + 8$

Note that these lines all have the same X intercept, but the slope of each one is twice the slope of the previous one. Look at the right triangles these lines make with the X and Y axes. Do the sides along the Y axis double as the slope doubles?

Exercise 21-6. Draw the graphs of

1. $Y = X + 2$
2. $Y = 3X + 6$

What about these right triangles?

Exercise 21-7. Graph $Y = 5$ to give a reference line, then graph

1. $Y = 3X$
2. $Y = \frac{3}{2}X$
3. $Y = \frac{3}{4}X$

Note the effect the slope has on the size of the triangles these lines make with the Y axis and the line $Y = 5$.

Exercise 21-8. Graph $Y = 5$ as a reference line, then

1. $Y = 2X$
2. $Y = X$
3. $Y = 0.5X$

Examine the size of the triangles again.

Exercises 21-4 through 21-8 show that doubling the slope of a line doubles its rate of rise or fall. Now it's time to refine this program. You have probably noticed that the apparent slopes of the lines are not right. For example, the line $Y = X$ does not make 45° angles with the axes. To make this obvious, draw the lines $Y = X$ and $Y = 5$, and compare the lengths of the legs of the right triangle.

Where did this trouble with angles come from? We started with a square in the X-Y plane determined by the inequalities $-10 \le X \le 10$ and $-10 \le Y \le 10$. Then we mapped this square onto the graphics screen. But the graphics screen is not square! It is longer from left to right than it is from top to bottom, so it's no wonder the angles aren't correct. How can we get the angles right? If we're going to keep the interval $-10 \le X \le 10$ for X, we'll have to use a shorter interval for Y. To find out what interval is appropriate for Y, experiment with LINES, as in the following exercise.

Exercise 21-9. Modify LINES so that you specify the Y bounds:

1. Change lines 410 and 420 so that the Y bounds are $YL = -B$ and $YH = B$.
2. Insert a line so you can input the value for B.
3. Experiment with this version of LINES to find the value for B that gives the correct apparent slopes.
4. Once you have found the best value for B, modify LINES so that it uses this value without your having to input it every time.

We will be working with several programs where having correct angles is important, so let's look at this scaling problem some more. By taking $YL = -B$, and $YH = B$, we have a rectangle with vertical dimension

$$YH - YL = 2B$$

and horizontal dimension

$$XH - XL = 20$$

So, the vertical-to-horizontal ratio is

$$\left(\frac{YH - YL}{XH - XL}\right) = \frac{2B}{20} = \frac{B}{10}$$

Any rectangle with this same ratio will give correct apparent slopes to lines. We use the ratio $S = B/10$ in subsequent programs. Different monitors require different values of S to give correct angles, so you must be able to set S for your monitor. Hence, use the value of B you found in Exercise 21-9 to compute the scale factor S for your monitor. Write this value of S down so you can refer to it later.

In the following exercises, use your scaled version of LINES from Exercise 21-9 and your knowledge of slopes of lines.

Exercise 21-10.

1. Draw a parallelogram with two sides horizontal.
2. Draw a parallelogram with no side horizontal.

3. Draw a parallelogram with one vertex at (0, 4).
4. Draw a parallelogram with one vertex at (0, 4) and one at (0, −3).
5. Draw a parallelogram with one vertex at (0, 4) and one at (5, 0).

Exercise 21-11.

1. Draw a rectangle with one side along the line Y = X.
2. Draw a rectangle with one side along the line Y = 2X + 3.
3. Draw a rectangle with one side along the line Y = 3X + 4 and one vertex at (0, −3).
4. Draw a rectangle with one vertex at (−2, 3) and one vertex at (3, 4).

In case you'd like to expand your straight-line graphing options, try the programming projects in the next two exercises.

Exercise 21-12. Write a program to graph a line given a point on the line and the slope of the line. One way to do this is to make the computer figure out the Y intercept of the line and then use the graphing procedure of LINES.

Exercise 21-13. Write a program to graph a line, given two points on it. You can use your solution to Exercise 21-12 here. If the desired line is not vertical, compute its slope and then use your program of Exercise 21-12.

22

GRAPHS

When studying a function Y = F(X), a lot of information can be obtained from its graph. For example, you may be able to see where the function is rising and falling, where it has its highest and lowest points, where it crosses the X and Y axes, and so forth. The most obvious way to sketch the graph is to use the formula Y = F(X) to compute several points (X, Y) on the graph and then to plot these points on graph paper. However, this process is often quite tedious to do by hand and may be nearly impossible for a complicated function. Fortunately, we can use the plotter subroutine discussed in Chapter 21, ''Lines,'' to make the computer draw graphs of functions for us.

The program GRAPHER (see Listing 22-1) uses this plotter subroutine to plot points on the graph of a general function. Before analyzing the program in detail, try it out.

Exercise 22-1. Run GRAPHER. As a first trial, use the function it comes with, Y = SIN(X)*COS(3*X) and graph from X = −3 to X = 3.

GRAPHER gives you the opportunity to change the function to be graphed. After you have the function you want, you specify the endpoints XL and XH of the interval of X values over which you want to see the graph. Then the program computes Y = F(X) at 140 points from XL to XH. It uses arrays to store the X and Y coordinates of each point. As it computes the X and Y coordinates, the program keeps a record of the highest and lowest Y values and their corresponding X coordinates. When the arrays of X and Y coordinates are complete, it uses the lowest Y value YL and the highest Y value YH to set the interval of Y values for the plotting subroutine.

Now let's look at the details of the program.

LISTING 22-1

```
100  REM     GRAPHER
110  GOTO 150
120  REM   SUBROUTINE TO COMPUTE Y FROM X:
130 Y =  SIN (X) *  COS (3 * X)
140  RETURN
150  HOME
160  PRINT "      THIS PROGRAM GRAPHS A FUNCTION."
170  PRINT : PRINT "YOU SUPPLY X COORDINATES OF THE LEFT-"
180  PRINT : PRINT "AND RIGHT-HAND EDGES OF THE SCREEN."
190  PRINT : PRINT "      THE Y COORDINATES OF THE TOP"
200  PRINT : PRINT "AND BOTTOM EDGES OF THE SCREEN"
210  PRINT : PRINT "ARE SET TO THE HIGHEST AND LOWEST"
220  PRINT : PRINT "LOWEST FUNCTION VALUES."
230  VTAB 23
240  PRINT "PRESS ANY KEY TO CONTINUE. "
250  GET ANS$
260  HOME
270  PRINT "      THE FUNCTION TO BE GRAPHED IS"
280  PRINT : PRINT "SPECIFIED IN LINE 130:"
290  LIST 130
300  PRINT : PRINT "DO YOU WISH TO CHANGE THE FUNCTION "
310  PRINT : PRINT "(Y OR N)";
320  INPUT ANS$
330  IF ANS$ <  > "Y" THEN 390
340  HOME
350  PRINT "ENDING FOR NOW SO YOU CAN CHANGE"
360  PRINT : PRINT "LINE 130."
370  LIST 130
380  END
390  HOME
400  PRINT : PRINT "    ENTER THE X BOUNDS"
410  PRINT : PRINT "FOR THE SCREEN:"
420  PRINT : PRINT "LOW X = ";: INPUT XL
430  PRINT : PRINT "HIGH X = ";: INPUT XH
440 NUM = 140: REM      NUMBER OF X STEPS
450  REM  SET UP ARRAY FOR X,Y COORDINATES
460  DIM X(NUM),Y(NUM)
470  REM     COMPUTE X STEP VALUE:
480 XS = (XH - XL) / NUM: REM     NUM STEPS XL TO XH
490  REM  INITIALIZE HIGH AND LOW Y:
500 X = XL
510  GOSUB 120: REM   COMPUTE Y
520 YH = Y:YL = Y:XU = X:XD = X
530  HOME
540  VTAB 10: HTAB 17
550  PRINT "COMPUTING"
560  REM  COMPUTE X,Y PTS ON GRAPH:
570  FOR K = 1 TO NUM
580 X(K) = XL + K * XS
590 X = X(K): GOSUB 120: REM    COMPUTE Y
600 Y(K) = Y
610  REM    UPDATE HIGH POINT:
620  IF Y(K) > YH THEN YH = Y(K):XU = X(K)
630  REM    UPDATE LOW POINT:
640  IF Y(K) < YL THEN YL = Y(K):XD = X(K)
650  NEXT
660  HOME
670  VTAB 21
680  LIST 130: REM   DISPLAY FUNCTION
690  VTAB 22
700  PRINT "   ": REM   WIPE OFF LINE NUMBER
710  REM  SET SCREEN FOR GRAPHICS:
```

```
720  HGR : HCOLOR= 3
730  REM  NOW PLOT THE POINTS:
740  FOR K = 1 TO NUM
750 X = X(K)
760 Y = Y(K)
770  GOSUB 890: REM    PLOT X,Y
780  NEXT
790  PRINT "PRESS ANY KEY TO CONTINUE.";
800  GET ANS$
810  HOME
820  VTAB 21
830  PRINT : PRINT "HIGH Y = ";YH;" AT X = ";XU
840  PRINT "LOW Y = ";YL;" AT X = ";XD
850  PRINT "PRESS ANY KEY TO CLEAR AND END.";
860  GET ANS$
870  HGR : TEXT : HOME : END
880  REM
890  REM  PLOTTER SUBROUTINE
900  REM
910  REM   CONVERT X,Y TO U,V:
920  REM
930  REM         X=XL TO U=0
940  REM          X=XH TO U=279
950  REM
960  REM          Y=YL TO V=159
970  REM         Y=YH TO V=0
980  REM
990 U = 279 * (X - XL) / (XH - XL)
1000 V =  - 159 * (Y - YH) / (YH - YL)
1010  REM  IF U OR V OUT OF RANGE, SKIP PLOT:
1020  IF U < 0 OR U > 279 THEN 1050
1030  IF V < 0 OR V > 159 THEN 1050
1040  HPLOT U,V
1050  RETURN
```

Lines 120 to 140 contain the subroutine that computes the value Y of the function for a given X value.

Lines 270 to 330 give you the chance to change the function. Note how line 290 uses the LIST command to display the function.

Lines 400 to 430 allow you to set the X interval.

Lines 440 to 460 set up the arrays for the X and Y coordinates. I used 140 X points rather than the maximum 280 to save time in computing. You can plot more points if you want by changing the value given to NUM in line 440.

Line 480 computes the X step size XS by dividing the total interval length, XH − XL, by the number of steps, NUM.

Lines 490 to 520 initialize the high and low Y values to the Y value at X = XL.

The FOR-NEXT loop in lines 570 to 650 computes the X and Y coordinates and the high and low Y values.

Line 580 uses the step size XS to generate the X values by adding a multiple of XS to the left-hand end point XL.

Lines 590 to 600 compute Y for the current X value.

Lines 610 to 640 update the high and low Y values as necessary.

Once the FOR-NEXT loop has computed all the X-Y values, lines 670 to 700 print the equation of the function being graphed.

Lines 710 to 720 put the computer in graphics mode.

Finally, lines 730 to 780 use a FOR-NEXT loop to plot the points on the screen, using the plotter subroutine, which resides in lines 890 to 1050.

After the plotting has finished, there is a pause in line 800, then lines 830 to 840 report the high and low points on the graph.

Now let's get back to using the program.

Exercise 22-2. Draw the graph of Y = sin(X) cos (3x) over the X intervals

1. $-1 \le X \le 1$
2. $-5 \le X \le 5$
3. $-10 \le X \le 10$
4. $-20 \le X \le 20$

As you can see from these graphs, the choice of the X interval can have a profound effect on the graph's appearance. If interval is too large or too small, you may get no useful picture at all. Of course, what is useful depends on what you are trying to find out about the function. You may have to experiment with various X-intervals to get one that suits you.

The current version of GRAPHER is rather restricted: You can't put in the X and Y axes, for example. So, let's upgrade the program.

Exercise 22-3. Add to GRAPHER the plotting of horizontal lines. We'll put the horizontal line plotter in at the end of the current program. Thus,

1. Change line 850 to

```
850 GO TO 1200
```

2. Add lines:
3. Run this version of GRAPHER, and put in the X axis.

LISTING 22-2

```
1200  PRINT "DO YOU WANT TO DRAW A HORIZONTAL LINE";
1210  INPUT ANS$
1220  IF ANS$ = "N" THEN 1280
1230  PRINT "HORIZONTAL LINE AT Y = ";
1240  INPUT Y
1250  FOR X = XL TO XH STEP XS
1260  GOSUB 890: REM     PLOT X,Y
1270  NEXT
1280  PRINT "ALL DONE";
1290  INPUT ANS$
1300  IF ANS$ = "N" THEN 1200
1310  TEXT : HOME : END
```

How about putting in the Y axis? To do this, you need to be able to draw vertical lines.

Exercise 22-4. Add to the current version of GRAPHER the plotting of vertical lines:

1. Delete the ending lines 1280 to 1310.
2. Insert new lines, starting with 1280:

```
1280 PRINT "PLOT VERTICAL LINE";
1290 INPUT ANS$
.
.
.
```

   Mimic the previous plotting of horizontal lines. You will need a Y step size YS, computed from YL and YH as XS was in line 350. V goes from 0 to 159, so 160 steps from YL to YH might be appropriate.
3. After the drawing of a vertical line, give the user the option of drawing more horizontal or vertical lines. See lines 1280 to 1310 of Listing 22-2 for guidance.
4. Run this version of GRAPHER and put in the X and Y axes.

Exercise 22-5. Draw the graph of $Y = X^2$ on the intervals

1. $-1 \leq X \leq 1$
2. $-5 \leq X \leq 5$
3. $-10 \leq X \leq 10$

Which interval gives the best overall picture of the function X^2?

Exercise 22-6. Draw the graph of $Y = X^3$ on the intervals

1. $-1 \leq X \leq 1$
2. $-5 \leq X \leq 5$
3. $-10 \leq X \leq 10$

Which interval gives the best overall picture of this function?

GRAPHER reports the highest and lowest function values for the X values it has considered in the interval $XL \leq X \leq XH$. Hence, it is giving an approximation to the maximum and minimum values for F(X) on that interval. Why only an approximation? Well, the computer can look only at a finite number of X points, so it might miss the X values that give the absolute largest and smallest values. However, if the step size XS is small, it shouldn't miss the true values by much.

You can also use GRAPHER to find *local maximum* points and *local minimum* points. The next exercise illustrates what is meant by local maximum and minimum points.

Exercise 22-7. Draw the graph of

$$Y = 2X^3 + 3X^2 - 6X$$

over the interval $-3 \le X \le 3$. Put in the X and Y axes.

The program reports the high point as $Y = 63$ at $X = 3$ and the low point as $Y = -9$ at $X = -3$. But you can see that the curve reaches a momentary high point between $X = -3$ and $X = 0$ and a momentary low point between $X = 0$ and $X = 3$. These are examples of local maximum and minimum points.

Exercise 22-8. Find the coordinates of the local maximum point for the function of Exercise 22-7. You can do this by making a suitable choice for the X interval for GRAPHER. You saw in the previous exercise that the function has a local maximum somewhere between $X = -3$ and $X = 0$. Thus, take these X values as the ends of the X interval for the graphing, and GRAPHER will report to you the coordinates of the local maximum.

Exercise 22-9. Find the coordinates of the local minimum point for the function of Exercise 22-7. Since you know the local minimum occurs between $X = 0$ and $X = 3$, you can use these points as end points of the X interval.

Exercise 22-10. For the function

$$Y = 2X^3 + 9X^2 - 60X + 2$$

use your expanded version of GRAPHER to

1. find the local maximum points
2. find the local minimum points
3. find the X coordinates where $Y = 0$

Exercise 22-11. Draw the graph of each of the following functions. In each case, choose an X interval so that the graph displays the important features of the function. (You have to decide what the important features are!)

1. $Y = (X - 1)(X - 2)(X - 3)$
2. $Y = \sin(X)$
3. $Y = \tan(X)$
4. $Y = X*\sin(X)$
5. $Y = e^X$

Exercise 22-12. Draw the graph of the function

$$Y = \begin{cases} \dfrac{\sin(X)}{X} & \text{if } X \neq 0 \\ 0.5 & \text{if } X = 0 \end{cases}$$

for $-2 \leq X \leq 2$.

This function has a separate definition for X = 0, so you must use an IF statement in the subroutine that computes Y. Because of the unusual treatment given the point X = 0, examine the behavior of the function carefully for X near 0, that is, draw the graph over short X intervals containing 0.

You see that the graph of this function has a break at X = 0. Can you redefine the function at X = 0 in order to eliminate the break?

23

VELOCITY

You are undoubtedly familiar with computer games in which objects move around on the screen. In this chapter we study motion in the plane and see how to use computer graphics to model such motion.

First we need to see how the velocity of a point affects its coordinates. Suppose a point is moving in the X-Y plane so that its velocity in the X direction is VX = 50 feet per second. If the X coordinate at a certain time is X = 7, then 0.1 second later the X coordinate is

$$X = 7 + 0.1*50 = 12$$

If the velocity had been VX = 100 feet per second, the X coordinate 0.1 seconds later would have been

$$X = 7 + 0.1*100 = 17$$

In general, with velocity VX, the new X coordinate 0.1 seconds later is

$$X = 7 + 0.1*VX$$

Since 7 was the old X coordinate, in BASIC, the change from the old X to the new X is written

$$X = X + 0.1*VX$$

We could, of course, compute the new X coordinate after any time interval DT, not just after 0.1 second. Then the new X coordinate is

$$X = X + DT*VX \tag{23-1}$$

This formula shows how to compute the new X coordinate given the old coordinate X, the time interval DT, and the X velocity VX.

The formula for motion in the Y direction is entirely similar. If the velocity in the Y direction is VY and the old Y coordinate is Y, the new Y coordinate after DT seconds is

$$Y = Y + DT*VY \tag{23-2}$$

The program VELOCITY uses formulas (23-1) and (23-2) to compute new X and Y coordinates for a point moving around in the X-Y plane.

Exercise 23-1. Run VELOCITY to see what it does.

LISTING 23-1

```
100  REM      VELOCITY
110  HOME
120  PRINT "     THIS PROGRAM DEMONSTRATES CONSTANT"
130  PRINT : PRINT "VELOCITY IN THE X-Y PLANE."
140  VTAB 7: PRINT "     YOU WILL SPECIFY THE INITIAL "
150  PRINT : PRINT "X AND Y COORDINATES AND THE INITIAL"
160  PRINT : PRINT "X AND Y VELOCITIES."
170  VTAB 16: PRINT "     THERE IS NO ACCELERATION IN"
180  VTAB 18: PRINT "THE X OR Y DIRECTIONS."
190  VTAB 23: PRINT "PRESS ANY KEY TO CONTINUE."
200  GET ANS$
210  REM   SET TIME INCREMENT:
220 DT = .1
230  REM  SET X BOUNDS:
240 XL =  - 100:XH = 100
250  REM   SET Y COORD FOR SCREEN CENTER:
260 YC = 0
270 S = .625: REM    SCALE FACTOR
280  REM  COMPUTE LOW Y AND HIGH Y
290  REM  EQUIDISTANT FROM YC AND SO
300  REM  THAT (YH-YL)/(XH-XL)=S:
310 YH = YC + S * (XH - XL) / 2
320 YL = YC - S * (XH - XL) / 2
330  HGR
340  VTAB 23
350  PRINT "ENTER INITIAL X COORDINATE"
360  PRINT : PRINT XL;" <= X <= ";XH
370  INPUT XI
380  PRINT : PRINT "ENTER INITIAL Y COORDINATE"
390  PRINT : PRINT YL;" <= Y <= ";YH
400  INPUT YI
410  REM  PLOT INITIAL LOCATION:
420  HCOLOR= 3
430 X = XI:Y = YI: GOSUB 670: REM   PLOT
440  PRINT "WHAT INITIAL X VELOCITY DO YOU WANT";
450  INPUT VX
460  PRINT : PRINT "WHAT INITIAL Y VELOCITY DO YOU WANT";
470  INPUT VY
```

```
480  REM  IF POINT IS OFF SCREEN, END:
490  IF X < XL OR X > XH THEN 600
500  IF Y < YL OR Y > YH THEN 600
510  REM  WIPE OUT OLD LOCATION:
520  HCOLOR= 0: GOSUB 670
530  REM   COMPUTE NEW X AND Y  COORDINATES:
540 X = X + DT * VX
550 Y = Y + DT * VY
560  REM  PLOT NEW LOCATION:
570  HCOLOR= 3
580  GOSUB 670: REM  PLOTTER
590  GOTO 490
600  HOME : VTAB 24
610  PRINT "OFF SCREEN."
620  PRINT "DO ANOTHER";
630  INPUT ANS$
640  IF ANS$ = "Y" THEN 330
650  TEXT : HOME : END
660  REM
670  REM  PLOTTER SUBROUTINE
680  REM
690  REM  CONVERT X,Y TO U,V:
700  REM
710  REM   X=XL TO U=0
720  REM   X=XH TO U=279
730  REM
740  REM   Y=YL TO V=159
750  REM  Y=YH TO V=0
760  REM
770 U = 279 * (X - XL) / (XH - XL)
780 V =  - 159 * (Y - YH) / (YH - YL)
790  REM     IF U OR V OUT OF RANGE, RETURN
800  IF U < 0 OR U > 279 THEN 830
810  IF V < 0 OR V > 159 THEN 830
820  HPLOT U,V
830  RETURN
```

Now let's analyze VELOCITY (see Listing 23-1).

Lines 210 to 220 set the time increment DT.

Lines 230 and 240 set the X bounds for the graphics screen.

Line 260 sets the Y coordinate of the center of the screen.

Lines 270 to 320 set the scale factor S and the Y coordinates of the top and bottom of the screen. The given value for S, 0.625, gives correct angles on my Apple monitor. In Chapter 21, you found the correct value of S for your monitor. If it is different from 0.625, change the value of S in line 270 to your value.

Line 330 sets the screen for graphics.

Lines 350 to 400 obtain the starting X and Y coordinates.

Lines 410 to 430 plot the starting location using the plotter subroutine, lines 660 to 830, which we discussed in Chapter 21, "Lines."

Lines 440 to 470 obtain the X and Y velocities.

Lines 480 to 500 check whether the point has coordinates that will put it on the screen. If not, the plotting ends.

Lines 510 to 520 erase the old point by replotting it in black (HCOLOR = 0).

Lines 530 to 550 compute new X and Y coordinates by formulas (23-1) and (23-2).

Lines 560 to 580 plot the new location.

This computing and plotting is repeated (see line 590) as long as the point is on the screen.

Now let's see how to get horizontal and vertical motions.

Exercise 23-2. Start at X = 0, Y = 0, and use velocities

1. VX = 70, VY = 0
2. VX = −70, VY = 0
3. VX = 0, VY = 70
4. VX = 0, VY = −70

Note the effect of negative velocities on the direction the point moves.

Exercise 23-3. Start at X = 0, Y = 0, and use velocities that make the point move

1. vertically upward
2. vertically downward
3. horizontally to the right
4. horizontally to the left

Now investigate the effect of the magnitude of the velocities.

Exercise 23-4. Start at X = −100, Y = 0, and use velocities

1. VX = 50, VY = 0
2. VX = 100, VY = 0
3. VX = 200, VY = 0

Notice the speed of the point: The larger VX is, the faster the point moves.

Exercise 23-5. Start at X = 0, Y = 55, and use velocities

1. VX = 0, VY = −20
2. VX = 0, VY = −40
3. VX = 0, VY = −80

From this exercise, you see that the speed of the point depends on the absolute value of VY.

Exercise 23-6. Start at X = 0, Y = 0, and use velocities that make the point move

1. slowly to the left

2. rapidly downward
3. slowly upward
4. rapidly to the right

Now try motion in a diagonal direction in Exercise 23-7.

Exercise 23-7. Start at X = −100, Y = −55, and use velocities

1. VX = 50, VY = 50
2. VX = 100, VY = 100
3. VX = 200, VY = 200

Exercise 23-8. Start at X = 100, Y = 55, and give velocities that will make it move

1. slowly down and to the left
2. rapidly down and to the left

To get some more information about the motion, it helps to see the track of the point.

Exercise 23-9. Disable the unplotting of old locations by changing line 520 to make it a REM statement:

```
520 REM HCOLOR = 0: GOSUB 670
```

Run this version a few times to see the paths. Use various starting locations and speeds. Note that the paths are straight lines.

Exercise 23-10. To be able to compare one path with another, change the program so that it doesn't wipe off the old path when asked to do another. Chance line 640 to

```
640 IF ANS$="Y" THEN 340
```

Try out this version on some motions of your choice.

Exercise 23-11. Run this version of VELOCITY to see the paths generated by

1. X = −100, y = −55, VX = 70, VY = 70
2. X = −50, Y = −55, VX = 70, VY = 70
3. X = 0, Y = −55, VX = 50, VY = 50

What geometric property do these lines satisfy?

Exercise 23-12. Run this version to see the paths generated by

1. $X = -100$, $Y = 55$, $VX = 30$, $VY = -60$
2. $X = -50$, $Y = 55$, $VX = 40$, $VY = -80$
3. $X = 0$, $Y = 55$, $VX = 50$, $VY = -100$

You see that these paths are parallel lines. That means that they all have the same slope. How is the slope determined b the velocities VX and VY?

If the point has coordinates (X, Y) and velocities VX and VY, then one second later, it has coordinates (X + VX, Y + VY), as shown in Figure 23-1. You see from the triangle that the slope of the line from the old point to the new one is VY/VX, so this is how the slope of the path is determined by the velocities.

What about the speed of the point? Over our one-second interval, the point moves a distance D equal to length of the hypotenuse of the triangle. Since the point moves this distance D each second, its speed is

$$\mathrm{SQR}(VX^2 + VY^2)$$

units per second.

The next two exercises involve the slopes of the paths.

Exercise 23-13.

1. Draw the path generated by starting at $X = 0$, $Y = 0$ and using $VX = 20$, $VY = 10$.
2. Start again at $X = 0$, $Y = 0$, but use velocities so that the new path has twice the slope of the old one.

Exercise 23-14.

1. Plot the path generated by $X = 100$, $Y = 55$, $VX = -30$, $VY = -50$.
2. Start at $X = -100$, $Y = 55$ and choose VX, VY so that the paths meet at right angles.

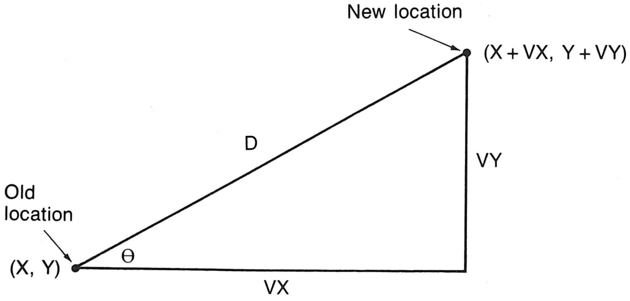


Figure 23-1 X and Y velocities

Thus you have seen that constant X and Y velocites generate straight-line paths and how the X and Y velocities give the slope of the path and the speed of the moving point. Now we go back to the original version of the program.

Exercise 23-15. Restore the unplotting of the old point. Change line 520 back to

```
520 HCOLOR = 0: GOSUB 670
```

For a different kind of motion, make the moving point bounce off the edges of the screen. How can we achieve the bouncing? Suppose the point is moving in a horizontal line with VX = 3. The velocity being positive means the point is moving to the right. When it reaches the right-hand edge of the screen, send it back to the left by changing the X velocity to VX = −3.

Similarly, if the X-velocity is VX = −3, when the point reaches the left-hand edge of the screen, change the X velocity to VX = 3. In general, when the point reaches the right- or left-hand edges, set VX = −1*VX. Similarly, to bounce off the top or bottom edge, let VY = −1*VY when the point reaches the top or bottom edge.

Exercise 23-16. Change the program in order to include bounces, as follows.

1. Remove the ending lines 600 to 650.
2. Insert new lines:

```
600 IF X < XL or X > XH THEN VX = -1*VX
610 IF Y < YL or Y > YH THEN VY = -1*VY
620 GO TO 510
```

Now that we have removed the ending lines of the old program, you will have to use the RESET button or CTRL-C to stop the plotting.

Exercise 23-17. Run this bouncy version for various starting locations and velocities. This kind of motion should remind you of various computer games.

Exercise 23-18. Disable the unplotting again to see the paths. Run this bouncy path version for various starting locatons and velocities. See what kind of patterns you can get.

24

ACCELERATION

In the Chapter 23, ''Velocity,'' we studied the motion of a point moving with constant velocity. We saw that the constant velocity gives straight-line paths. Now we move on to consider acceleration. We again use the computer's graphics screen to picture the motion.

First we need to see how to compute the X and Y coordinates of a point undergoing some acceleration. Because the point's velocity is changing, this situation is more complicated than that described in Chapter 23.

Suppose our point is moving in the X direction with velocity VX = 10 feet per second. Suppose also that the acceleration in the X direction is AX = 30 feet per second per second. This means that in every second, the velocity increases by 30 feet per second. So, after 0.1 second, the velocity is

$$VX = 10 + 0.1*30 = 13$$

feet per second. After DT seconds, the new velcoity is

$$VX = 10 + DT*30$$

feet per second. In general, for starting velocity VX and constant acceleration AX, the velocity after DT seconds is

$$VX + DT*AX$$

so, in BASIC,

```
VX = VX + DT*AX
```

computes the new X velocity at the end of DT seconds.

Now that we know how to compute the new X velocity, let's see how to find the new X coordinate. There is a bit of trouble in computing the new X location because the velocity is not constant. If the velocity VX were constant, the new X coordinate would be computed from the old by

```
X = X + DT*VX
```

Here, however, the velocity is not constant because of the acceleration. If we set up program lines

```
xxx X = X + DT*VX

yyy VX = VX + DT*AX
```

then line xxx computes the new X coordinate as if the X velocity were constantly equal to the old X velocity VX. Line yyy then computes the new X velocity. Although line xxx cannot give the correct value for X, it should give a good approximation to the new X location, that is, provided that the time increment DT is small enough that the velocity is approximately constant over the time interval.

In a similar fashion, we can compute the new Y location and velocity. Given the old values Y and VY for the Y coordinate and Y velocity, if the Y acceleration is AY, we will use program lines

```
www Y = Y + DT*VY

zzz VY = VY + DT*AY
```

Again, line www computes an approximation to the correct Y value by pretending that the Y velocity is constantly equal to its value at the beginning of the time interval.

Although you could have acclerations in both the X and Y directions, let's start with acceleration only in the Y direction. We will mimic gravity at the earth's surface by taking AY to be constant and negative (so that a point moving upward will lose speed). We take AX to be zero so that the only acceleration is downward.

The program ACCELERATION (see Listing 24-1) uses the formulas noted above and is quite similar to the program VELOCITY, which we discussed in the preceding chapter.

LISTING 24-1

```
100  REM     ACCELERATION
110  HOME
120  PRINT "      THIS PROGRAM DEMONSTRATES CONSTANT"
130  PRINT : PRINT "ACCELERATION IN THE X-Y PLANE."
140  VTAB 8: PRINT "     YOU WILL SPECIFY:"
150  VTAB 11: PRINT "       INITIAL X AND Y COORDINATES"
160  VTAB 13: PRINT "       INITIAL X AND Y VELOCITIES"
170  VTAB 16: PRINT "     THERE IS A CONSTANT NEGATIVE"
180  PRINT : PRINT "ACCELERATION IN THE Y DIRECTION "
190  PRINT : PRINT "(AS IN GRAVITY) AND NO X ACCELERATION"
200  VTAB 23: PRINT "PRESS ANY KEY TO CONTINUE."
210  GET ANS$
220  REM   SET TIME INCREMENT:
230 DT = .1
240  REM  SET X BOUNDS:
250 XL =  - 100:XH = 100
260  REM    SET Y COORD FOR SCREEN BOTTOM:
270 YL = 0
280 S = .625: REM   SCALE FACTOR
290  REM   COMPUTE HIGH Y SO THAT
300  REM     (YH-YL)/(XH-XL)=S:
310 YH = S * (XH - XL)
320  HGR
330  VTAB 23
340  PRINT "ENTER INITIAL X COORDINATE"
350  PRINT : PRINT XL;" <= X <= ";XH
360  INPUT XI
370  PRINT : PRINT "ENTER INITIAL Y COORDINATE"
380  PRINT : PRINT YL;" <= Y <= ";YH
390  INPUT YI
400  REM  PLOT INITIAL LOCATION:
410  HCOLOR= 3
420 X = XI:Y = YI: GOSUB 760: REM   PLOT
430  PRINT "WHAT INITIAL X VELOCITY DO YOU WANT";
440  INPUT VX
450  PRINT : PRINT "WHAT INITIAL Y VELOCITY DO YOU WANT";
460  INPUT VY
470  HOME : VTAB 23
480  PRINT "INITIAL CONDITIONS: ";
490  PRINT "X = ";XI;",  Y = ";YI
500  PRINT "        X VEL = ";VX;",  Y VEL = ";VY
510  REM  SET X AND Y ACCELERATIONS:
520 AX = 0
530 AY =  - 32: REM  FEET PER SEC PER SEC
540  REM  WIPE OUT OLD LOCATION:
550  HCOLOR= 0: GOSUB 760
560  REM    COMPUTE NEW X COORDINATE:
570 X = X + DT * VX
580  REM    IF X OUT OF RANGE, END:
590  IF X < XL OR X > XH THEN 710
600  REM  COMPUTE NEW Y COORDINATE:
610 Y = Y + DT * VY
620  REM  IF Y OUT OF RANGE, END:
630  IF Y < YL THEN 710
640  REM  COMPUTE NEW X AND Y VELOCITIES:
650 VX = VX + DT * AX
660 VY = VY + DT * AY
670  REM  PLOT NEW LOCATION:
680  HCOLOR= 3
690  GOSUB 760: REM  PLOTTER
700  GOTO 540
710  REM   HERE POINT IS OFF SCREEN
```

```
720  PRINT "OFF SCREEN."
730  PRINT "DO ANOTHER (Y OR N) ";: INPUT ANS$
740  IF ANS$ <  > "N" THEN 320
750  TEXT : HOME : END
760  REM  PLOTTER SUBROUTINE
770  REM
780  REM  CONVERT X,Y TO U,V:
790  REM
800  REM   X=XL TO U=0
810  REM   X=XH TO U=279
820  REM
830  REM   Y=YL TO V=159
840  REM  Y=YH TO V=0
850  REM
860 U = 279 * (X - XL) / (XH - XL)
870 V =  - 159 * (Y - YH) / (YH - YL)
880  REM    IF U OR V OUT OF RANGE, RETURN:
890  IF U < 0 OR U > 279 THEN 920
900  IF V < 0 OR V > 159 THEN 920
910  HPLOT U,V
920  RETURN
930  REM
```

Line 230 sets the time interval to 0.1 second.

Lines 240 to 310 set the X and Y bounds for the graphics screen. The bottom of the screen is set to Y = 0, and the top is adjusted to give the correct proportions.

Lines 340 to 420 obtain and plot the initial location.

Lines 430 to 460 get the initial X and Y velocities.

Lines 470 to 500 print the initial conditions beneath the graphics screen.

Lines 520 to 530 set the acceleration values.

Line 550 erases the old point from the graphics screen.

Line 570 computes the new X coordinate.

Line 590 tests whether the X coordinate is within the screen bounds. If not, the plotting ends.

Line 610 computes the new Y coordinate.

Line 630 tests whether the point is below the bottom edge of the screen. If it is, the plotting ends.

Lines 640 to 660 compute new X and Y velocities.

Finally, lines 670 to 690 plot the new location.

Exercise 24-1. Use ACCELERATION to model the motion starting at X = 0, Y = 100, VX = 0, VY = 0. This models the action of dropping a rock from a height of 100 feet.

Exercise 24-2. Observe the motion starting with X = −100, Y = 100, VX = 60, VY = 0. Here you're throwing a rock horizontally from a height of 100 feet.

Exercise 24-3. See what happens if you start with X = 0, Y = 0, VX = 0, VY = 80. Now you're throwing a ball straight up.

Exercise 24-4. Try out various motions starting off from the ground (start with Y = 0) and going up at an angle.

Exercise 24-5. Disable the erasing of old points so you can see the paths as follows:

1. Change line 550 to a remark statement:

   ```
   550 REM HCOLOR = 0: GOSUB 760
   ```

2. Do Exercise 1-5 again, and see what kind of paths arise. Do you get familiar types of curves?

   Now put some bounce in the action as you did with the straight-line motion in Chapter 23.

Exercise 24-6. Make the point bounce off the ground by changing line 630 to

```
630 IF Y ≤ YL THEN VY = −1*VY: GO TO 610
```

Run this version of the program for various starting locations and velocities. Now put even more bounce into it.

Exercise 24-7. Make the point bounce off the walls by changing line 590 to

```
590 IF X ≤ XL OR X ≤ XH THEN VX = -1*VX: GO TO 570
```

Try this version out for several starting conditions.

We have been specifying the X and Y velocities separately. You could instead specify the starting speed and the starting angle with respect to the X axis. If the starting speed is SPD and the angle is THETA, then

$$\mathrm{VX} = \mathrm{SPD}*\cos(\mathrm{THETA})$$

$$\mathrm{VY} = \mathrm{SPD}*\sin(\mathrm{THETA})$$

To see this, look at Figure 23-1. In one second, the distance traveled is

$$\mathrm{D} = \mathrm{SPD}*1 = \mathrm{SPD}$$

so SPD is the length of the hypotenuse of the triangle. Similarly, in one second, the X distance traveled is VX and the Y distance is VY, so VX and VY are the legs of the triangle. Hence the formulas above for VX and VY follow from the definitions of sine and cosine.

Exercise 24-8. Modify the original program ACCELERATION so that you specify the starting speed and angle, rather than VX and VY. Since the computer's SIN and COS functions expect radian angles, you will have to convert from degrees to radians if you want to input your angles in degrees.

Exercise 24-9. Run this speed-angle version for motions starting at X = 0, Y = 0, with

1. speed = 50, angle = 90°
2. speed = 50, angle = 30°
3. speed = 50, angle = 60°
4. speed = 50, angle = 120°
5. speed = 50, angle = 150°

Exercise 24-10. Make the program report the coordinates of the point when it leaves the screen. Then start at X = −100, Y = 0, with speed = 70, and find the angle that will make the point go the farthest before it hits the ground.

The final three exercises give suggestions for further investigations of motion in the plane.

Exercise 24-11. So far we've had acceleration only in the Y direction. Now insert a constant acceleration in the X direction. This models the situation where gravity acts in the Y direction and a constant wind blows in the X direction. Note that the paths you get now are no longer parabolas.

Exercise 24-12. How about trying a nonconstant acceleration? For example, you could have a wind that blows stronger at higher altitudes than at lower ones. To model this, you need an X acceleration AX which is an increasing function of Y.

Exercise 24-13. Our bouncing has been rather idealistic so far in that the point loses no speed when it bounces. Try making the point lose some velocity with each bounce.

25

KEPLER AND NEWTON

In Chapters 23 and 24, we studied motion in the X-Y plane under constant velocity and constant acceleration. Here we turn our attention to a famous motion problem involving nonconstant acceleration—the motion of the planets around the sun. As in the previous chapters, we use the graphics screen of the computer to picture the orbiting planets.

By studying data from astronomical observations, Johannes Kepler found three laws that the planets obey:

Kepler's first law: The path of each planet about the sun is an ellipse with the sun at one focus.

Kepler's second law: Each planet moves so that the line from the planet to the sun sweeps out equal areas in equal times.

Kepler's third law: The orbits of the planets are such that the square of the time required to complete one orbit is proportional to the cube of the mean distance from the planet to the sun.

Kepler published the first two discoveries in 1609 and the third in 1618. For centuries, scientists had attempted to explain the motions they saw the planets making in terms of circles, which they considered to be the most perfect curves. Such explanations were always very complicated and Kepler's ellipses provided a much simpler description.

In 1687, Isaac Newton found an even simpler foundation for the planetary motions. He found a single law that implied (and was implied by) Kepler's three laws. Newton's law of gravity states that the force of the sun upon a planet acts in the direction of the sun. The magnitude of the force is proportional to the product of the masses of the sun and the planet and inversely proportional to the square of the distance between them.

This law of gravity gives a formula for the force upon a planet. From Newton's equation relating force and acceleration

$$\mathbf{F} = m\mathbf{A} \tag{25-1}$$

(where **F** is the force vector, **A** the acceleration vector, and m the mass of the planet), we get a formula for the acceleration of the planet. Once we have the acceleration, we can approximate the velocity of the planet, and from the velocity, the position. We then plot the motion of the planet on the screen and see if Kepler's laws seem to be true.

Let's get the equations with which to model the motion. We'll let the sun be located at the origin (0, 0) and the planet at the point (X, Y). Then the distance from the planet to the sun is

$$R = \sqrt{X^2 + Y^2} \tag{25-2}$$

Let the mass of the planet be m and of the sun be M. By Newton's law, the force on the planet is directed toward the sun. From equation (25-1), so also must be the acceleration **A**. Hence,

$$\mathbf{A} = (-kX, -kY) \tag{25-3}$$

for some positive scalar k. From this equation, the magnitude of the acceleration is

$$|\mathbf{A}| = \sqrt{k^2X^2 + k^2Y^2}$$

$$k\sqrt{X^2 + Y^2}$$

by algebra

$$= kR$$

from equation (25-2). Thus,

$$|\mathbf{A}| = kR \tag{25-4}$$

But the second part of Newton's law of gravity says that the magnitude of the force is

$$|\mathbf{F}| = \frac{cmM}{R^2} \tag{25-5}$$

for some constant c. So, from equation (25-1),

$$|\mathbf{A}| = \frac{|\mathbf{F}|}{m}$$

$$= \frac{cM}{R^2}$$

by equation (25-5). Hence, from equation (25-4),

$$kR = \frac{cM}{R^2}$$

so that

$$k = \frac{cM}{R^3} \tag{25-6}$$

Finally, then, equation (25-3) gives

$$\mathbf{A} = \left(\frac{-cMX}{R^3}, \frac{-cMY}{R^3}\right) \tag{25-7}$$

To make things simple, we assume that the units are chosen so that $c = 1$, so that the acceleration is

$$\mathbf{A} = \left(\frac{-MX}{R^3}, \frac{-MY}{R^3}\right) \tag{25-8}$$

This says that the X acceleration at a point (X, Y) is

$$AX = \frac{-M*X}{R^3} \tag{25-9}$$

and the Y acceleration is

$$AY = \frac{-M*Y}{R^3} \tag{25-10}$$

We use these formulas for acceleration to model the motion of the planet.

Let the (old) location of the planet be (XO, YO). Let the X velocity be HO and the Y velocity be VO. (H is for "horizontal"; V is for "vertical.") We want to approximate the (new) location (XN, YN) and velocities HN, VN for the planet after a certain amount DT of time has elapsed.

Even though the X and Y velocities are changing throughout the time interval, we can approximate the new X and Y coordinates by using the old X and Y velocities:

$$XN = XO + DT*HO$$
$$YN = YO + DT*VO$$

(These equations would give the exact coordinates if the X and Y velocities were constant throughout the interval. We'll take a small time interval DT and hope they give a good approximation to the true coordinate values.)

Now we'll do a similar approximation for the velocities. Even though the X and Y accelerations are changing throughout the time interval, we can approximate the new X and Y velocities by

$$HN = HO - \frac{DT*M*XO}{R^3}$$

$$VN = VO - \frac{DT*M*YO}{R^3}$$

Here we're using the acceleration as given by equations (25-9) and (25-10) using the old X and Y coordinates.

The program KEP 1 (see Listing 25-1) uses the method above to approximate the motion of a planet.

LISTING 25-1

```
100  REM    KEP 1
110  GOTO 390
120  REM         PLOTTER SUBROUTINE
130  REM  CONVERT X,Y TO U,V:
140  REM
150  REM         X=XL  TO U=0
160  REM         X=XH  TO U=279
170  REM
180  REM         Y=YL  TO V=159
190  REM         Y=YH  TO V=0
200  REM
210 U = 279 * (X - XL) / (XH - XL)
220 V =  - 159 * (Y - YH) / (YH - YL)
230  REM  IF OFF SCREEN, RETURN:
240  IF U < 0 OR U > 279 THEN 270
250  IF V < 0 OR V > 159 THEN 270
260  HPLOT U,V
270  RETURN
280  REM        SUBROUTINE TO UPDATE VARIABLES:
290  REM  COMPUTE NEW X,Y COORDINATES:
300 XN = XO + DT * HO
310 YN = YO + DT * VO
320  REM  COMPUTE NEW DISTANCE TO SUN:
330 RN =  SQR (XN ^ 2 + YN ^ 2)
340  REM  COMPUTE NEW HORIZ AND VERT VELOCITIES:
350  REM  NEWTON'S LAW OF GRAVITY!
360 HN = HO - M * DT * XO / (RO ^ 3)
370 VN = VO - M * DT * YO / (RO ^ 3)
380  RETURN
390  HOME
400  PRINT "    THIS PROGRAM ILLUSTRATES KEPLER'S"
410  PRINT : PRINT "FIRST LAW OF PLANETARY MOTION:"
420  PRINT : PRINT : PRINT
430  PRINT "     THE ORBIT OF A PLANET IS AN ELLIPSE"
440  PRINT "WITH THE SUN AT ONE FOCUS."
450  VTAB 20
460  PRINT "PRESS ANY KEY TO CONTINUE."
```

```
470  GET ANS$
480  HOME
490  PRINT "      YOU SUPPLY INITIAL X,Y LOCATIONS"
500  PRINT : PRINT "AND INITIAL X AND Y VELOCITIES"
510  PRINT : PRINT "    NEW X AND Y LOCATIONS ARE COMPUTED"
520  PRINT : PRINT "USING NEWTON'S LAW OF GRAVITY."
530  PRINT : PRINT "     THE LOCATION IS THEN PLOTTED."
540  VTAB 20
550  PRINT "PRESS ANY KEY TO CONTINUE."
560  GET ANS$
570  REM  SET CONSTANTS:
580 M = 750: REM  MASS OF SUN
590 DT = .1: REM  TIME INCREMENT
600  REM  SET X,Y BOUNDS FOR SCREEN:
610 S = .625: REM    SCALE FACTOR FOR SCREEN PROPORTIONS
620  REM  SET LOW AND HIGH X:
630 XL =  - 20:XH = 20
640  REM  SET Y COORD OF VERTICAL CENTER OF SCREEN:
650 YC = 0
660  REM   GET YL,YH SYMMENTRIC ABOUT YC
670  REM   SO THAT (YH-YL)/(XH-XL)=S:
680 YL = YC - S * (XH - XL) / 2
690 YH = YC + S * (XH - XL) / 2
700  HOME
710  PRINT "ENTER STARTING X COORDINATE,"
720  PRINT : PRINT XL;" <= X <= ";XH;" :"
730  PRINT : PRINT "X = ";: INPUT XI
740  PRINT : PRINT "ENTER STARTING Y COORDINATE,"
750  PRINT : PRINT YL;" <= Y <= ";YH;" :"
760  PRINT : PRINT "Y = ";: INPUT YI
770  PRINT : PRINT "STARTING X VELOCITY";: INPUT HI: REM  H="H
ORIZONTAL"
780  PRINT : PRINT "STARTING Y VELOCITY";: INPUT VI: REM  V
= "VERTICAL"
790  HGR : HCOLOR= 3
800  REM  PLOT SUN:
810 X = 0:Y = 0: GOSUB 120
820  VTAB 23
830  PRINT "STARTING CONDITIONS:"
840  PRINT "X = ";XI;"  Y = ";YI
850  PRINT "X VEL = ";HI;"  Y VEL = ";VI
860  REM  INTITIALIZE VARIABLES:
870 RI =  SQR (XI ^ 2 + YI ^ 2): REM  RADIAL DISTANCE TO SUN
880 XO = XI:YO = YI: REM  OLD X,Y
890 HO = HI:VO = VI: REM  OLD HORIZ, VERT VELOCITIES
900 RO = RI: REM  OLD RADIAL DISTANCE
910  REM         MAIN PROGRAM:
920  REM  PLOT LOCATION XO,YO:
930 X = XO:Y = YO
940  GOSUB 120: REM  PLOT X,Y
950  REM  COMPUTE NEW LOCATION AND VELOCITIES:
960  GOSUB 280: REM   XO,YO,HO,VO,RO  IN, XN,YN,HN,VN ,RN OUT
970  REM  RESET VARIABLES FOR NEXT COMPUTATION:
980 XO = XN:YO = YN
990 RO = RN
1000 HO = HN:VO = VN
1010  GOTO 910: REM  DO IT AGAIN
```

Now let's analyze KEP 1.

Lines 120 to 270 contain the familiar plotting subroutine.

Lines 280 to 380 contain a subroutine to compute a new location and new velocities as described above.

Lines 400 to 550 contain a greeting.

Lines 570 to 590 set the mass of the sun and the time interval length.

Lines 600 to 690 contain the familiar setup of X and Y bounds for plotting. If the correct scale factor for your monitor is different from 0.625, change the value of S in line 610 to the value you found in Chapter 21.

Lines 710 to 780 obtain the initial location and velocities.

Line 790 puts the computer in graphics mode.

Lines 800 and 810 plot the sun.

Lines 830 to 850 print the starting location and velocities.

Lines 860 to 900 initialize the variables.

Lines 920 to 940 plot the current point on the screen.

Line 960 calls the subroutine that computes a new point and velocities.

Lines 970 to 1000 update the variables for the next round.

Exercise 25-1. Run KEP 1 for various starting situations. Do you get ellipses for orbits?

What a disappointment—the planets refuse to follow nice elliptical orbits. So were Kepler and Newton wrong? Not likely, you say, and you're right. The trouble lies in our program, not their theories. Let's see how to improve the program.

Lines 300 and 310 compute the new location (XN, YN) based on the velocities at the old point (XO, YO). Then lines 360 and 370 compute new velocities based on the acceleration at the old point. Since the acceleration is not constant, it makes just as much sense to compute the new velocities using the acceleration at the *new* point.

Exercise 25-2. Change lines 360 and 370 to use the acceleration at the new point:

```
370 HN = HO − M*DT*XN/RN³
380 HN = HO − M*DT*YN/RN³
```

Run this version for various starting locations and velocities. Do you like the orbits better?

The improvement in the orbits given by Exercise 25-2 shows that altering an approximation technique may change the results significantly. The fact that you get elliptical orbits shows that Newton's law may indeed imply Kepler's first law.

You have probably noticed a couple of things about this version of the program: First, if the planet gets too close to the sun, it may not give a nice closed ellipse for an orbit. This is because the velocity computations in lines 360 and 370 involve division by the radial distance. If this distance is close to zero, it may cause significant round-off error. Second, if the planet leaves the screen, the program reports nothing about the planet. It will be helpful to at least know the planet's coordinates when it is off the screen.

Exercise 25-3. Insert some lines between 910 and 920 so that the program will print the coordinates XO and YO if (and only if!) the point is off the screen.

The next few exercises ask you to use your current version of KEP 1 to investigate some of the properties of the orbits.

Exercise 25-4. Start at X = 10, Y = 0, with X velocity = 0. Find a Y velocity that will give a circular orbit.

Exercise 25-5. Repeat Exercise 25-4 for various starting X coordinates. Try to find a formula for the circular orbit velocity in terms of the distance of the starting point from the sun.

Exercise 25-6. Start at X = 10, Y = 0, with X velocity = 0. Find the largest Y velocity for which the planet will stay in orbit around the sun. (That is, from that location, what is the escape velocity?)

Now that you have seen the elliptical orbits of Kepler's first law, let's see if his second law seems to hold. His second law says that the radial line from the sun to the planet sweeps out equal areas in equal times. This means that the planet will have to move faster when it is closer to the sun. Have you noticed that in the orbits you've already seen?

To test Kepler's second law, we must be able to compute the area swept out by the planet's radial line in a given amount of time. One way to approximate this area is to add up the areas of the triangles made up by the sun and the planet's old and new locations. Since we can easily compute the sides of this triangle (the old and new radial distances and the distance the planet has moved), we need a formula for the area of a triangle in terms of its side lengths. What we need is *Heron's formula:*

$$A = \sqrt{s(s - a)(s - b)(s - c)}$$

for the area A of the triangle with sides a, b, and c. In this formula, $s = (a + b + c)/2$ is the semiperimeter of the triangle.

In case you're not familiar with Heron's formula, the program HERON'S DEMO (see Listing 25-2) compares triangle areas computed by the base-height formula and Heron's formula so that you can see they give the same results.

LISTING 25-2

```
100  REM    HERON'S DEMO
110  HOME
120  PRINT "      THIS PROGRAM DEMONSTRATES"
130  PRINT : PRINT "THE EQUIVALENCE OF TWO FORMULAS"
140  PRINT : PRINT "FOR THE AREA OF A TRIANGLE:"
150  PRINT : PRINT : PRINT "BASE-HEIGHT FORMULA:  AREA=BASE*HE
IGHT/2"
160  PRINT : PRINT "HERON'S FORMULA:"
170  PRINT : PRINT "     AREA=SQR(S*(S-S1)*(S-S2)*(S-S3))  "
180  PRINT : PRINT "WHERE S=SEMI-PERIMETER AND THE"
190  PRINT : PRINT "LENGTHS OF THE SIDES ARE S1,S2,S3."
200  VTAB 23
210  PRINT "PRESS ANY KEY TO CONTINUE."
220  GET ANS$
230  HOME
240  PRINT "ENTER THE X COORDINATES OF TWO POINTS"
250  PRINT : PRINT "ON THE X-AXIS:"
260  PRINT : PRINT "FOR FIRST POINT, X = ";
270  INPUT X1
```

```
280  PRINT : PRINT "FOR SECOND POINT, X = ";
290  INPUT X2
300  HOME
310  PRINT "ENTER X AND Y COORDINATES FOR THIRD "
320  PRINT : PRINT "VERTEX:
330  PRINT : PRINT "X = ";: INPUT X3
340  PRINT : PRINT "Y = ";: INPUT Y3
350 ALT =  ABS (Y3)
360 BASE =  ABS (X1 - X2)
370 S1 =  SQR ((X2 - X3) ^ 2 + (Y2 - Y3) ^ 2)
380 S2 =  SQR ((X1 - X3) ^ 2 + (Y1 - Y3) ^ 2)
390 S3 =  SQR ((X1 - X2) ^ 2 + (Y1 - Y2) ^ 2)
400 SEMI = (S1 + S2 + S3) / 2
410 HNS =  SQR (SEMI * (SEMI - S1) * (SEMI - S2) * (SEMI - S3)
)
420  HOME
430  PRINT "      FOR POINTS"
440  PRINT : PRINT "(";X1;",0)"
450  PRINT : PRINT "(";X2;",0)"
460  PRINT : PRINT "(";X3;",";Y3;")"
470  PRINT : PRINT "ALTITUDE TO X-AXIS BASE:";ALT
480  PRINT : PRINT "LENGTH OF BASE :";BASE
490  PRINT : PRINT "HALF BASE TIMES HEIGHT: ";ALT * BASE / 2
500  PRINT : PRINT "LENGTHS OF SIDES:"
510  PRINT : PRINT S1;"  ";S2;"   ";S3
520  PRINT : PRINT "HERON'S FORMULA GIVES:";HNS
```

Exercise 25-7. Analyze and run HERON'S DEMO.
Now let's get back to Kepler's second law.

Exercise 25-8. Modify your current version of KEP 1 to investigate Kepler's second law. The general plan is as follows: After each new location is found, compute the area of the triangle formed by the sun and the old and new locations of the planet. Keep a running total of these areas and print the total after every block of 10 steps. Here are detailed suggestions for the programming:

Let the variable T denote the elapsed time, AREA the area of each triangle, and AS the running sum of the triangle areas. After each new location is computed, compute

1. The new time value T = T + DT.
2. The distance DIST the planet traveled in this step from the coordinates (XO, YO), (XN, YN) of the old and new points.
3. The semiperimeter SEMI of the triangle, from DIST and the new and old radial distances, RN and RO.
4. The area AREA of the triangle, by Heron's formula.
5. Updated area sum AS = AS + AREA.
6. Use a variable COUNTER to count the number of locations computed. Each time COUNTER is a multiple of 10, print the time T, the coordinates XN, YN, and the area sum AS for the last 10 steps. Then clear the graphics screen, reset the area sum AS to 0, and do another 10 steps. Run this version of the program. Does Kepler's second law seem to hold?

Now let's investigate Kepler's third law. It says the square of the time for one orbit is proportional to the cube of the mean radial distance. This means, no matter what size orbit you have, if you compute the time for the orbit T and the mean radius RM, then the quotient

$$\frac{T^2}{RM^3}$$

is always the same.

Exercise 25-9. Modify KEP 1 again, this time to investigate Kepler's third law. Here, you must find the time required for one orbit. How can you tell when one orbit has been completed? One thing you *cannot* do is to wait for the coordinates XN, YN to equal the starting values XI, YI, because it's unlikely that XN and YN will ever again exactly match XI and YI. One way to see when an orbit has been completed is to start with $Y = 0$ and watch for Y to change sign. When Y changes sign the first time, the planet is halfway around the sun. When Y changes sign for the second time, you may declare the orbit to be complete.

To find the mean distance to the sun, find the largest and smallest values of the radial distance and take their average. Use variables RL and RH to record the low and high values for the radial distance and TL and TH to record the corresponding times.

After each new location has been computed,

1. Update RL and RH if necessary.
2. Check to see whether the Y coordinate has changed sign.

When Y changes sign for the second time, print:

1. Initial location and velocities
2. Low R value and corresponding time
3. High R value and corresponding time
4. Mean radius $RM = (RH + RL)/2$
5. Time T for orbit
6. Kepler's number: T^2/RM^3

Then end. Run this version for various orbits. Does Kepler's third law seem to hold?

26

PARAMETRIC EQUATIONS

In Chapters 23 and 24, "Velocity" and "Acceleration," respectively, we modeled motion in the plane by approximating the X and Y coordinates of the moving point based on certain assumptions about its velocity or acceleration. Another way to prescribe motion is to make the X and Y coordinates of a point change according to specified functions:

$$\begin{aligned} X &= F(T) \\ Y &= G(T) \end{aligned} \tag{26-1}$$

Such equations are called *parametric equations*. The variable T is the *parameter*, and we often think of T as representing time. By choosing the functions F and G in the equations suitably, you can make the point (X, Y) travel on a particular path with a desired speed and direction.

The program PARAMETRIC EQUATIONS (see Listing 26-1) plots the path of a point moving according to parametric equations. First, let's get acquainted with what the program does.

Exercise 26-1. Run PARAMETRIC EQUATIONS for the equations it comes with. Take the low X bound to be -2 and the high X bound to be 2. Plot 30 steps from time $T = 0$ to time $T = 3$.

LISTING 26-1

```
100  REM       PARAMETRIC EQUATIONS
110  HOME
120  PRINT "     THIS PROGRAM DRAWS THE PATH OF A "
```

```
130  PRINT : PRINT "POINT MOVING ACCORDING TO PARAMETRIC"
140  PRINT : PRINT "EQUATIONS"
150  PRINT : PRINT "            X = F(T)"
160  PRINT : PRINT "            Y = G(T)"
170  VTAB 23
180  PRINT "      PRESS ANY KEY TO CONTINUE."
190  GET ANS$
200  HOME
210  PRINT : PRINT "    YOU WILL SUPPLY THE X BOUNDS FOR "
220  PRINT : PRINT "THE SCREEN.  THE Y COORDINATE OF THE"
230  PRINT : PRINT "CENTER OF THE SCREEN IS SET TO 0."
240  PRINT : PRINT "     HIGH AND LOW Y BOUNDS ARE THEN"
250  PRINT : PRINT "COMPUTED WHICH WILL GIVE CORRECT"
260  PRINT : PRINT "PROPORTIONS TO THE GRAPHS."
270  PRINT : PRINT "     YOU WILL ALSO SPECIFY THE NUMBER"
280  PRINT : PRINT "OF TIME VALUES TO BE USED IN THE"
290  PRINT : PRINT "PLOTTING."
300  VTAB (23): PRINT "     PRESS ANY KEY TO CONTINUE."
310  GET ANS$
320  HOME
330  PRINT : PRINT "     THE EQUATIONS ARE SPECIFIED IN "
340  PRINT : PRINT "LINES "
350  PRINT
360  LIST 810
370  LIST 820
380  PRINT
390  PRINT : PRINT "DO YOU WISH TO CHANGE THE EQUATIONS"
400  PRINT : PRINT "( Y OR N )";
410  INPUT ANS$
420  IF ANS$ <  > "Y" THEN 1030
430  HOME
440  PRINT "ENDING NOW SO YOU CAN CHANGE"
450  PRINT : PRINT "THE EQUATIONS."
460  LIST 810,820
470  PRINT : PRINT "TO RESTART, TYPE 'RUN 1030'"
480  END
490  REM  SUBROUTINE TO SET SCREEN BOUNDS
500  HOME
510  PRINT : PRINT "    ENTER THE X BOUNDS FOR THE SCREEN:"
520  PRINT : PRINT
530  PRINT : PRINT "LOW X = ";: INPUT XL
540  PRINT : PRINT "HIGH X = ";: INPUT XH
550 YC = 0: REM   CENTRAL Y COORDINATE
560 S = .625: REM     SCALE FACTOR
570  REM  COMPUTE LOW Y AND HIGH Y
580  REM  EQUIDISTANT FROM YC AND SO
590  REM  THAT (YH-YL)/(XH-XL)=S:
600 YH = YC + S * (XH - XL) / 2
610 YL = YC - S * (XH - XL) / 2
620  PRINT : PRINT "LOW Y = ";YL;", HIGH Y = ";YH
630  RETURN
640  REM  SUBROUTINE TO SET TIME INTERVAL
650  PRINT : PRINT "STARTING TIME = ";
660  INPUT TL
670  PRINT : PRINT "ENDING TIME = ";
680  INPUT TH
690  PRINT : PRINT "NUMBER OF STEPS TO BE PLOTTED = ";: INPUT
BOUND
700 TS = (TH - TL) / BOUND: REM    TIME STEP SIZE
710  RETURN
720  REM  SUBROUTINE TO DRAW CURVE
730  HGR : HCOLOR= 3
740  REM   DISPLAY FUNCTION FORMULA:
750  HOME : VTAB (21)
760  LIST 810,820
770  VTAB (21)
```

```
780  PRINT "    ": PRINT "     ": REM  WIPE OFF LINE NUMBERS
790  PRINT "    T FROM ";TL;" TO ";TH;", ";BOUND;" STEPS"
800  FOR T = TL TO TH STEP TS
810 X =  COS (T)
820 Y =  SIN (T)
830  GOSUB 860: REM  PLOT X,Y
840  NEXT
850  RETURN
860  REM  PLOTTER SUBROUTINE
870  REM
880  REM  CONVERT X,Y TO U,V:
890  REM
900  REM   X=XL TO U=0
910  REM   X=XH TO U=279
920  REM
930  REM   Y=YL TO V=159
940  REM    Y=YH TO V=0
950  REM
960 U = 279 * (X - XL) / (XH - XL)
970 V =  - 159 * (Y - YH) / (YH - YL)
980  REM  IF U OR V OUT OF RANGE, SKIP PLOT:
990  IF U < 0 OR U > 279 THEN 1020
1000  IF V < 0 OR V > 159 THEN 1020
1010  HPLOT U,V
1020  RETURN
1030  REM  MAIN PROGRAM
1040  GOSUB 490: REM  SET SCREEN BOUNDS
1050  GOSUB 640: REM  SET TIME INTERVAL
1060  GOSUB 720: REM  PLOT CURVE
1070  PRINT "DO ANOTHER (Y OR N)";
1080  INPUT ANS$
1090  IF ANS$ = "N" THEN  TEXT : HOME : END
1100  PRINT "CHANGE EQUATIONS";
1110  INPUT ANS$
1120  IF ANS$ = "Y" THEN  TEXT : GOTO 430
1130  PRINT "CHANGE SCREEN BOUNDS";
1140  INPUT ANS$
1150  IF ANS$ = "Y" THEN  GOSUB 530
1160  PRINT "CHANGE TIME INTERVAL OR NUMBER OF STEPS";
1170  INPUT ANS$
1180  IF ANS$ = "Y" THEN  GOSUB 650
1190  PRINT "READY TO PLOT";
1200  INPUT ANS$
1210  IF ANS$ = "Y" THEN  GOSUB 720
1220  GOTO 1070
```

Now let's analyze the program.

Lines 110 to 370 print a program description and show the equations currently specified in the program.

Lines 390 to 480 give you the opportunity to stop the program in case you want to change the equations. If you don't want to change the equations, line 420 transfers control to line 1030, the start of the main program.

Lines 490 to 630 contain a subroutine that sets the X-Y screen bounds. If your monitor requires a scale factor different from the given S = 0.625, change the value of S in line 560 to the value you found in Chapter 21.

Lines 640 to 710 contain a subroutine to set the starting and ending times, the number of points to be plotted, and the time step size.

Lines 720 to 850 are the subroutine that draws the curve. This subroutine calls the point-plotting subroutine (lines 860 to 1020) which you first met in Chapter 21, "Lines."

The main program, lines 1030 to 1220, calls the subroutines discussed above to plot the path of the given equations and then gives you the chance to change the setup for another plotting.

Use PARAMETRIC EQUATIONS in the following problems.

Exercise 26-2. Set the X bounds for the screen to $-2 \le X \le 2$, and plot 100 steps for the equations

$$X = \cos(T)$$
$$Y = \sin(T)$$

over the time intervals

1. $0 \le T \le 2\pi \approx 6.28$
2. $0 \le T \le \pi$
3. $\frac{\pi}{2} \le T \le \frac{3\pi}{2}$

Note the effect of the different ranges for the parameter. These parametric equations come from the unit circle definitions of sine and cosine. (See Figure 26-1.) So, for these equations, T represents the radian measure of the central angle which determines the point on the circle.

Exercise 26-3. Plot the equations of Exercise 26-1 again, this time with different screen bounds:

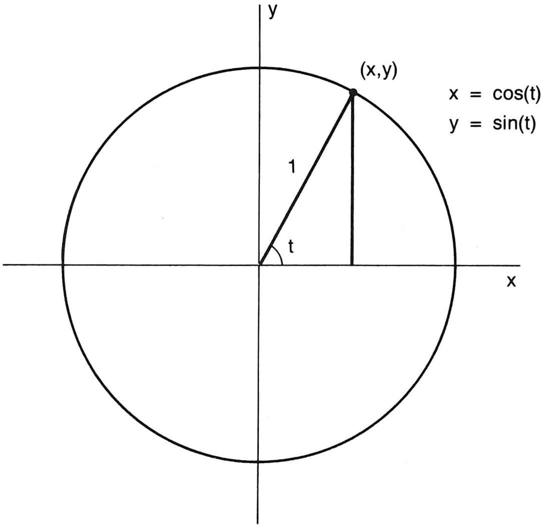


Figure 26-1 Unit circle

1. $-3 \leq X \leq 3$
2. $-4 \leq X \leq 4$
3. $-10 \leq X \leq 10$
4. $-1 \leq X \leq 1$

As you see from these plottings, the size of the picture depends on your choice of X bounds.

The equations we have been using so far,

$$X = \cos(T)$$

$$Y = \sin(T)$$

give a point moving on the unit circle because X and Y satisfy the equation of this circle:

$$X^2 + Y^2 = 1 \tag{26-2}$$

If fact, any functions $X = F(T)$ and $Y = G(T)$ that satisfy equation (26-2) give a point that moves on the unit circle. By choosing different functions F and G, you can get different circular motions.

Exercise 26-4. Plot 100 steps for the equations

$$X = \cos(2T)$$

$$Y = \sin(2T)$$

over the same time intervals as in Exercise 26-1. Compare with the motion of Exercise 26-1.

Exercise 26-5. Plot the motion given by

$$X = \cos(-T)$$

$$Y = \sin(-T)$$

and compare with the motions of Exercise 26-2.

You see from these last two exercises that doubling the parameter T doubles the speed of the moving point and that changing from T to $-T$ changes its direction.

Exercise 26-6. Find parametric equations that make the point trace the bottom half of the circle

1. in the counterclockwise direction
2. in the clockwise direction

Exercise 26-7. Consider the equations

$$X = \cos(T^2)$$

$$Y = \sin(T^2)$$

Since X and Y satisfy the equation of the unit circle, this point still moves on that circle. But now the angle is T^2, which is always positive and grows more rapidly than T did. Plot 50 steps for the equations, for

1. $0 \leq T \leq \pi$
2. $-1 \leq T \leq 2$
3. $0 \leq T \leq 2$

Compare your results with the previous circle motions.

Exercise 26-8. Plot $X = \sin(T)$, $Y = \cos(T)$. Compare with the previous circle motions.

Exercise 26-9. Plot 100 steps for each of the following. Try to predict in advance what sort of motion you will see.

1. $X = 0,\ Y = \sin(T),\quad 0 \leq T \leq 2$
2. $X = T,\ Y = \sin(T),\quad 0 \leq T \leq 2$
3. $X = -T,\ Y = \sin(T),\quad 0 \leq T \leq 2$
4. $X = T,\ Y = T,\quad -2 \leq T \leq 2$
5. $X = T,\ Y = -T,\quad -2 \leq T \leq 2$
6. $X = -T,\ Y = T,\quad -2 \leq T \leq 2$
7. $X = -T,\ Y = -T,\quad -2 \leq T \leq 2$

Exercise 26-10. Find parametric equations that make the point move along the line $Y = 2X-2$

1. up and to the right
2. down and to the left

Exercise 26-11. Find parametric equations that make the point move along the curve $Y = X^2$ so that it oscillates from $(-1, 1)$ to $(1, 1)$ and back again.

Exercise 26-12. Plot each of the following curves for $0 \leq T \leq 2\pi$.

1. $X = \sin(T),\ Y = \sin(T)$
2. $X = \sin(2T),\ Y = \sin(T)$

3. X = sin(T), Y = sin(2T)
4. X = sin(T), Y = sin(3T)
5. X = sin(2T), Y = sin(3T)

In general, for X = sin(mT), Y = sin(nT), describe how the curve's shape depends on m and n. (These are called *Lissajous* curves.)

Exercise 26-13. Plot X = 2*sin(T), Y = cos(T). Does this look like a familiar curve? Prove that it is: Use the identity

$$\sin^2(T) + \cos^2(T) = 1$$

to eliminate T from the equations.

Exercise 26-14. Plot X = sin(T), Y = cos(2T). Look familiar? Prove it.

27

POLAR COORDINATES

Most of the time we use the rectangular coordinates X and Y when we want to describe points in the plane. We usually stick with this system of coordinates because it is the first one we learn and because straight lines have very simple equations in it. However, there are other coordinate systems around, and each has its own advantages. The system of polar coordinates, which we study here, is particularly well suited for describing circular curves.

Rectangular coordinates are determined by distances to two lines (the X and Y axes). Polar coordinates, however, are determined by the distance to a point O, the *pole* of the system, and the angle to a ray, the *polar axis*. (See Figure 27-1). The polar coordinates (R, θ) of a point A in the plane are determined as follows: R is the distance from the point to the pole, and θ is the angle the segment OR makes with the polar axis.

We measure angles as positive in the counterclockwise direction and use radian measure. If the ray OA in Figure 27-1 rotates in the counterclockwise direction through 2 radians, it will return to its old location. This shows that the point A has more than one polar coordinate representation. Namely, it is represented by

$$(R, \theta), (R, \theta + 2\pi), (R, \theta + 4\pi)$$

and so forth. Moreover, if we use clockwise rotation, it is represented equally well by

$$(R, \theta - 2\pi), (R, \theta - 4\pi)$$

and so forth. So here is one startling difference between polar and rectangular coordinates: A point always has infinitely many polar coordinate representations, whereas it has a unique representation in rectangular coordinates.

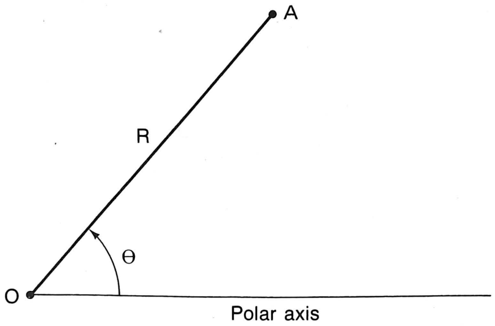


Figure 27-1 Polar axis and pole

Because the computer screen is set up in terms of rectangular coordinates, we must convert polar coordinates to rectangular ones to do our graphing. To make this easy, we take the pole to be the origin of the rectangular system and the polar axis to be the positive X axis. (See Figure 27-2.) From the right triangle in Figure 27-2, note that

$$\frac{X}{R} = \cos(\theta)$$

and

$$\frac{Y}{R} = \sin(\theta)$$

Thus, we have the following expressions for X and Y in terms of R and θ

$$X = R*\cos(\theta)$$
$$Y = R*\sin(\theta) \qquad (27\text{-}1)$$

The program POLAR POINT PLOTTER (see Listing 27-1) uses the pair of equations (27-1) and the usual plotter subroutine to draw points when you give their polar coordinates. Since BASIC doesn't recognize Greek symbols, we use T to stand for θ.

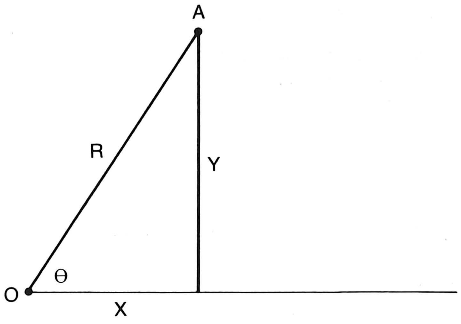


Figure 27-2 Polar triangle

LISTING 27-1

```
100  REM   POLAR POINT PLOTTER
110  HOME
120  PRINT "      THIS PROGRAM PLOTS POINTS WITH"
130  PRINT : PRINT "GIVEN POLAR COORDINATES (R,THETA)."
140  PRINT : PRINT "YOU SPECIFY THETA AS A MULTIPLE OF PI."
150  VTAB 23
160  PRINT "PRESS ANY KEY TO CONTINUE."
170  GET ANS$
180  HOME
190  REM  SET SCREEN BOUNDS:
200 XL =  - 10
210 XH = 10
220 YC = 0: REM   CENTRAL Y COORDINATE
230 S = .625: REM     SCALE FACTOR
240  REM  COMPUTE LOW Y AND HIGH Y
250  REM  EQUIDISTANT FROM YC AND SO
260  REM  THAT (YH-YL)/(XH-XL)=S:
270 YH = YC + S * (XH - XL) / 2
280 YL = YC - S * (XH - XL) / 2
290 PI = 4 *  ATN (1): REM   COMPUTER'S PI
300  HGR : HCOLOR= 3
310  REM  DRAW POLAR AXIS:
320 Y = 0
330  FOR X = 0 TO XH STEP XH / 140
340  GOSUB 490: REM  PLOT X,Y
350  NEXT
360  HOME : VTAB 21
370  PRINT "R = ";: INPUT R
380  PRINT "THETA = K*PI FOR K = ";: INPUT K
390 THETA = K * PI
400  REM     CONVERT (R,THETA) TO (X,Y):
410 X = R *  COS (THETA)
420 Y = R *  SIN (THETA)
430  IF X < XL OR X > XH THEN  PRINT "OFF SCREEN": GOTO 460
440  IF Y < YL OR Y > YH THEN  PRINT "OFF SCREEN": GOTO 460
450  GOSUB 490: REM  PLOT X,Y
460  PRINT "DO ANOTHER";: INPUT ANS$
470  IF ANS$ = "N" THEN  HGR : TEXT : HOME : END
480  GOTO 360
490  REM  PLOTTER SUBROUTINE
500  REM
510  REM  CONVERT X,Y TO U,V:
520  REM
530  REM   X=XL TO U=0
540  REM   X=XH TO U=279
550  REM
560  REM   Y=YL TO V=159
570  REM    Y=YH TO V=0
580  REM
590 U = 279 * (X - XL) / (XH - XL)
600 V =  - 159 * (Y - YH) / (YH - YL)
610  REM  IF U OR V OUT OF RANGE, SKIP PLOT:
620  IF U < 0 OR U > 279 THEN 650
630  IF V < 0 OR V > 159 THEN 650
640  HPLOT U,V
650  RETURN
```

First look at the program.

Lines 190 to 280 set the X and Y bounds for the screen in the usual fashion. If the scale factor you found for your monitor in Chapter 21 is different from the given S = 0.625, change S in line 230 to the value for your monitor.

Lines 290 and 300 set the value of PI and the graphics screen.

Lines 310 to 350 draw the polar axis.

Lines 360 to 390 obtain the polar coordinates of the point to be plotted. Because it is most convenient to think of radian angles as multiples of π, that's how the angle T is described in lines 380 and 390.

Lines 400 to 450 convert R and T to X and Y by the pair of equations (27-1) and plot the point if it is on the screen.

Lines 460 to 480 give you the option to plot another point.

Lines 490 to 650 contain the usual plotter subroutine.

Use POLAR POINT PLOTTER in the next few exercises to get some familiarity with polar coordinates.

Exercise 27-1. Plot the following points:

1. $R = 4, \theta = 0.5*PI$
2. $R = 4, \theta = PI$
3. $R = 4, \theta = 1.5*PI$

Before going on, note that these three points are all equidistant from the pole and that the angles put the points where you would expect: $\theta = \pi/2$ radians is 90° from the positive X axis, and the point in item 1 is straight up from the origin. $\theta = \pi$ radians is 180° from the positive X axis, and the point in item 2 is directly to the left of the origin.

Now add some more points:

4. $R = 4, \theta = 0.25*PI$
5. $R = 4, \theta = 0.75*PI$
6. $R = 4, \theta = 1.25*PI$
7. $R = 4, \theta = 1.75*PI$

These points come halfway around in each quadrant because $\pi/4$ radians = 45°. Now add even more points, each time using $R = 4$.

You see from this exercise that a circle with radius 4 and center at the pole has a very simple polar equation: $R = 4$.

In general, the circle with its center at the pole and radius equal to a has polar equation

$$R = a.$$

In rectangular coordinates, its equation is

$$X^2 + Y^2 = a^2$$

which is considerably more complicated than the polar version.

Exercise 27-2. Plot the points with R = 4 and each of the following angles. Note how the negative angles correspond to a clockwise rotation from the positive X axis.

1. $\theta = -0.25*\mathrm{PI}$
2. $\theta = -0.5*\mathrm{PI}$
3. $\theta = -0.75*\mathrm{PI}$
4. $\theta = -1*\mathrm{PI}$
5. $\theta = -1.25*\mathrm{PI}$
6. $\theta = -1.5*\mathrm{PI}$
7. $\theta = -1.75*\mathrm{PI}$

Exercise 27-3.

1. Use the angle $\theta = 0.25*\mathrm{PI}$ and plot the point with each of the following R coordinates: R = 7, 6, 5, 4, 3, 2, 1. Note that all these points lie on the line through the origin that makes a 45° angle with the polar axis.
2. Now continue with $\theta = 0.25*\mathrm{PI}$, and plot the points with the following R coordinates: $R = -1, -2, -3, -4, -5, -6, -7$. These points lie on the same line as those of part 1, but on the other side of the origin.

Exercise 27-4. Repeat Exercise 27-3, this time with $\theta = 0.66*\mathrm{PI}$.

Exercises 27-3 and 27-4 reveal two facts about polar coordinates. First, the polar equation $\theta = b$ gives a straight line through the origin. Second, the points (R, θ) and $(-R, \theta)$ lie on a line through the origin. Furthermore, they are the same distance from the pole but on opposite sides of it.

The next exercise gives more details about points with negative R coordinates.

Exercise 27-5.

1. Plot the point with $R = -4$, $\theta = 0.5*\mathrm{PI}$. Note where the point is. This is the same location you got in Exercise 27-1 with R = 4 and $\theta = 1.5*\mathrm{PI}$.
2. Plot the point with $R = -4$, $\theta = 0.25*\mathrm{PI}$. This is the same location you got in Exercise 27-1 with R = 4 and $\theta = 1.25*\mathrm{PI}$.

   Thus you see that there is even more ambiguity to polar coordinates than that of the multiple angles we discussed before. Here you see that the polar coordinates (−4, 0.5*PI) and (4, 1.5*PI) represent the same point, and so do (−4, 0.25*PI) and (+4,1.25*PI).

Exercise 27-6. For each of the following, plot the point with the given polar coordinates, then find a polar coordinate representation for the same point with R positive.

1. (−4, PI)
2. (−4, 1.5*PI)

3. (−4, 0.75*PI)
4. (−4, 1.25*PI)

As you can see from these examples, if a point has polar coordinates (R, θ) with R negative, it also has polar coordinates with $|R|$ as the first polar coordinate. If you switch to this positive first coordinate, how must you change θ?

Now that you have some experience with points in polar coordinates, we turn to functions. A polar coordinate function may be specified by an equation

$$R = F(\theta)$$

Such an equation gives the radial distance R as a function of the polar angle θ. So, as θ varies, the point moves closer to the origin or farther away depending on the function F. In the circle example we saw above, we had R = 4, so the function here is just a constant—the simplest possible function.

The program POLAR GRAPHER (see listing 27-2) graphs polar coordinate functions.

LISTING 27-2

```
100  REM            POLAR GRAPHER
110  HOME
120  PRINT "     THIS PROGRAM PLOTS THE GRAPH OF A "
130  PRINT : PRINT "POLAR COORDINATE FUNCTION R=F(THETA)."
140  PRINT : PRINT "    YOU WILL SUPPLY THE X BOUNDS FOR "
150  PRINT : PRINT "THE SCREEN.  THE Y COORDINATE OF THE"
160  PRINT : PRINT "CENTER OF THE SCREEN IS SET TO 0."
170  PRINT : PRINT "     HIGH AND LOW Y BOUNDS ARE THEN"
180  PRINT : PRINT "COMPUTED WHICH WILL GIVE CORRECT"
190  PRINT : PRINT "PROPORTIONS TO THE GRAPHS."
200  PRINT : PRINT "     YOU WILL ALSO SPECIFY THE NUMBER"
210  PRINT : PRINT "OF THETA VALUES TO BE USED IN THE"
220  PRINT : PRINT "PLOTTING."
230  VTAB (23): PRINT "PRESS ANY KEY TO CONTINUE."
240  GET ANS$
250  HOME
260  PRINT : PRINT "     THE FUNCTION IS SPECIFIED IN LINE"
270  PRINT : PRINT "840:"
280  PRINT
290  LIST 840
300  PRINT
310  PRINT : PRINT "DO YOU WISH TO CHANGE THE FUNCTION"
320  PRINT : PRINT "( Y OR N )";
330  INPUT ANS$
340  IF ANS$ <  > "Y" THEN 910
350  HOME
360  PRINT "ENDING NOW SO YOU CAN CHANGE"
370  LIST 840
380  PRINT "TO RESTART, TYPE ’RUN 910’"
390  END
400  REM  SUBROUTINE TO SET SCREEN BOUNDS
410  HOME
420  PRINT : PRINT "    ENTER THE X BOUNDS FOR THE SCREEN:"
430  PRINT : PRINT
440  PRINT : PRINT "LOW X = ";: INPUT XL
```

```
450  PRINT : PRINT "HIGH X = ";: INPUT XH
460 YC = 0: REM   CENTRAL Y COORDINATE
470 S = .625: REM      SCALE FACTOR
480  REM  COMPUTE LOW Y AND HIGH Y
490  REM  EQUIDISTANT FROM YC AND SO
500  REM  THAT (YH-YL)/(XH-XL)=S:
510 YH = YC + S * (XH - XL) / 2
520 YL = YC - S * (XH - XL) / 2
530  PRINT : HTAB 24: PRINT "LOW Y = ";YL
540  PRINT : HTAB 24: PRINT "HIGH Y = ";YH
550  VTAB 20
560  PRINT "PRESS ANY KEY TO CONTINUE."
570  GET ANS$
580  RETURN
590  REM  SBRTN TO SET THETA VALUES
600 PI = 4 *  ATN (1): REM  COMPUTER'S PI
610  HOME
620  PRINT "STARTING THETA = K*PI"
630  PRINT : PRINT "FOR WHAT K";: INPUT KL
640 TL = KL * PI: REM   LOW THETA
650  PRINT : PRINT "ENDING THETA = K*PI"
660  PRINT : PRINT "FOR WHAT K";: INPUT KH
670 TH = KH * PI: REM   HIGH THETA
680  PRINT : PRINT "NUMBER OF THETA STEPS = ";: INPUT BOUND
690 TS = (TH - TL) / BOUND: REM  THETA STEP SIZE
700  RETURN
710  REM  SBRTN TO GRAPH FUNCTION
720  HGR : HCOLOR= 3
730  REM   DISPLAY FUNCTION FORMULA:
740  HOME
750  VTAB (21)
760  LIST 840
770  VTAB (22)
780  PRINT "    ": REM  WIPE OFF LINE NUMBER
790  PRINT KL;"*PI <= T <= ";KH;"*PI"
800  REM  PLOT ORIGIN:
810 X = 0:Y = 0
820  GOSUB 1060
830  FOR T = TL TO TH STEP TS
840 R = 3 *  COS (T)
850  REM      CONVERT (R,THETA) TO (X,Y):
860 X = R *  COS (T)
870 Y = R *  SIN (T)
880  GOSUB 1060: REM  PLOT X,Y
890  NEXT
900  RETURN
910  REM  MAIN PROGRAM
920  GOSUB 400: REM   GET SCREEN BOUNDS
930  GOSUB 590: REM  GET THETA SET UP
940  GOSUB 710: REM  GRAPH FUNCTION
950  PRINT "DO ANOTHER ONE";: INPUT ANS$
960  IF ANS$ = "N" THEN  HGR : TEXT : HOME : END
970  PRINT "CHANGE FUNCTION";: INPUT ANS$
980  IF ANS$ = "Y" THEN  TEXT : GOTO 350
990  PRINT "CHANGE SCREEN BOUNDS";: INPUT ANS$
1000  IF ANS$ = "Y" THEN  TEXT : GOSUB 400
1010  PRINT "CHANGE THETA SET UP";: INPUT ANS$
1020  IF ANS$ = "Y" THEN  TEXT : GOSUB 590
1030  PRINT "READY TO PLOT";: INPUT ANS$
1040  IF ANS$ = "Y" THEN  GOSUB 710
1050  GOTO 950: REM  PRESENT OPTIONS AGAIN
1060  REM  PLOTTER SUBROUTINE
1070  REM
1080  REM  CONVERT X,Y TO U,V:
1090  REM
1100  REM    X=XL TO U=0
```

```
1110  REM    X=XH TO U=279
1120  REM
1130  REM    Y=YL TO V=159
1140  REM     Y=YH TO V=0
1150  REM
1160 U = 279 * (X - XL) / (XH - XL)
1170 V =   - 159 * (Y - YH) / (YH - YL)
1180  REM  IF U OR V OUT OF RANGE, SKIP PLOT:
1190  IF U < 0 OR U > 279 THEN 1220
1200  IF V < 0 OR V > 159 THEN 1220
1210  HPLOT U,V
1220  RETURN
```

This program is broken into subroutines so that you can repeat various parts of it without repeating the others.

Lines 400 to 580 are a subroutine that sets the X and Y bounds for the screen. Again, you may have to adjust the value of S in line 470 to suit your monitor.

Lines 590 to 700 are a subroutine that gets the starting and ending values for T and the number of points to plot.

Lines 710 to 900 are the graphing subroutine. Lines 750 to 790 display the function formula below the graphics screen. Line 840 contains the function equation. Lines 860 to 870 convert R and T to X and Y for the plotter subroutine.

Lines 910 to 1050 call the various subroutines and present you with options for changing the plotting after you have seen the graph.

Exercise 27-7. Get acquainted with POLAR GRAPHER by graphing the given function for various screen settings and θ arrangements.

Exercise 27-8. Graph the circle R = 3. Use 50 θ steps and graph for

1. $0 \leqslant \theta \leqslant 2\pi$
2. $\pi \leqslant \theta \leqslant 2\pi$
3. $-\pi/2 \leqslant \theta \leqslant \pi/2$
4. $2\pi \leqslant \theta \leqslant 3\pi$

Note how the choice of the θ domain affects the graph.

Because the function cos(T) oscillates between -1 and $+1$, it can be used to give some interesting curves. Some examples are in the next two exercises.

Exercise 27-9.

1. Graph the function $R = 1 - \cos(T)$, for $0 \leqslant \theta \leqslant 2\pi$. This curve is called a *cardioid.* Note how the values of cos(T) make the curve heart-shaped. At $T = 0$, $\cos(T) = 1$, so $R = 0$, and the point is at the pole. When $T = \pi/2$, $\cos(T) = 0$, so $R = 1$; when $T = \pi$, $\cos(T) = -1$, so $R = 2$; and when $T = 3\pi/2$, $\cos(T) = 0$, so $R = 1$ again.
2. Graph the cardioid $R = 1 - \cos(T + \pi/2)$. Note that the shape is the same as for the curve of part 1, but the orientation has changed.

3. Change the function so that you get a right-side-up heart.

Exercise 27-10. Graph each of the following:

1. $R = 1 - 2\cos(T)$
2. $R = 2 - \cos(T)$

Note the effect of the location of the 2 and the fact that cos(T) varies from -1 to 1:

In part 1, R becomes negative since 2*cos(T) goes all the way down to -2. Hence, the curve must pass through the pole. In part 2, R is never smaller than 1, so the curve is always one unit or more from the pole.

Exercise 27-11. Graph $R = \sin(T)$. By converting this equation to one in X and Y, prove that $R = \sin(T)$ gives a circle.

Exercise 27-12. Curves of the form $R = \sin(nT)$ are called *roses*. (After seeing the graphs, you may decide that ''daisies'' is a better name!) Graph the following roses. Note the number of petals and the range of T values required to trace the rose just once.

1. $R = \sin(2T)$
2. $R = \sin(3T)$

Graph more roses. For $R = \sin(nT)$, how many petals does the rose have? What range of T values traces the rose just once?

We turn now to an important application of polar coordinates, graphing the *conic section* curves—circles, ellipses, parabolas, and hyperbolas. You have seen above that a circle with center at the pole has a very simple equation in polar coordinates. Each of the other kinds of conic sections can be described in terms of a point O, the *focus*; a line L, the *directrix*; and a ratio ECC, the *eccentricity*. The curve is defined to be all those points A that satisfy the relation

$$d(A, O) = ECC*d(A, L) \qquad (27\text{-}2)$$

That is, the distance from the point A to the focus O is ECC times the distance from point A to the directrix L. (See Figure 27-3.) If the eccentricity ECC = 1, then the point A is equidistant from the focus and the directrix, and the curve is a *parabola*. If the eccentricity is less than 1, the curve is an *ellipse*. If the eccentricity is larger than 1, the curve is a *hyperbola*.

To find polar coordinate equations for these curves, we take the focus to be at the pole and the directrix to be a vertical line X = P. (See Figure 27-4.) If the point A has polar coordinates R and θ, you see from Figure 27-4 that d(A, O) = R, and

$$\begin{aligned} d(A, L) &= P - X \\ &= P - R*\cos(\theta) \end{aligned}$$

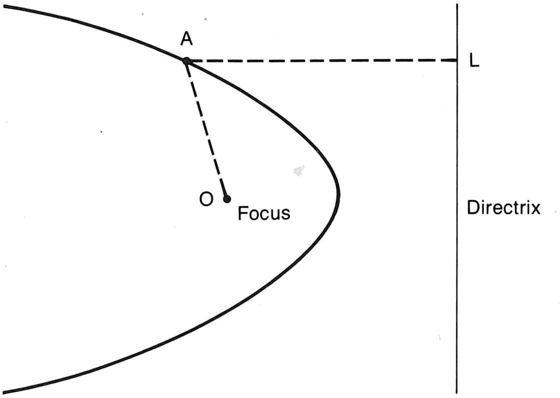

A
L
O
Focus
Directrix

Figure 27-3 Conic, focus, and directrix

Then the defining equation of the conic, equation (27-2), becomes

$$R = ECC^* [P - R^*\cos(\theta)]$$

Now we solve this for R: Multiply out the right-hand side to get

$$R = ECC^*P - R^*ECC^*\cos(\theta)$$

Add R*ECC*cos(θ) to both sides:

$$R + R^*ECC^*\cos(\theta) = ECC^*P$$

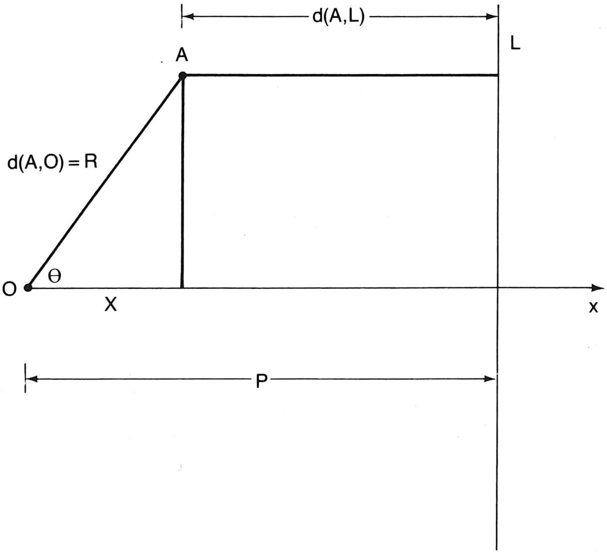

d(A,L)
L
A
d(A,O) = R
θ
O
X
x
P

Figure 27-4 Pole and directrix

Factor out R on left-hand side:

$$R*[1 + ECC*\cos(\theta)] = ECC*P$$

Finally, divide by 1 + ECC *cos(θ):

$$R = \frac{ECC*P}{1 + ECC*\cos(\theta)} \tag{27-3}$$

This is the polar equation of a conic section with the pole as focus, the line X = P as directrix, and ECC for eccentricity.

Now we'll draw some of these curves to see what effect the directrix and eccentricity have on them. The program POLAR CONICS (see Listing 27-3) graphs the curve after asking you for the directrix and eccentricity.

LISTING 27-3

```
100  REM    POLAR CONICS
110  HOME
120  PRINT "     THIS PROGRAM DRAWS THE GRAPH OF A"
130  PRINT : PRINT "CONIC SECTION USING POLAR COORDINATES."
140  PRINT : PRINT "     YOU SPECIFY THE ECCENTRICITY"
150  PRINT : PRINT "AND DIRECTRIX X = P."
160  PRINT : PRINT
170  PRINT : PRINT "ECCENTRICITY = ";: INPUT ECC
180  PRINT : PRINT "DIRECTRIX X = P, -10 <= P <= 10"
190  PRINT : PRINT "FOR P = ";: INPUT P
200 XL =  - 10
210 XH = 10
220 YC = 0: REM   CENTRAL Y COORDINATE
230 S = .625: REM    SCALE FACTOR
240  REM  COMPUTE LOW Y AND HIGH Y
250  REM  EQUIDISTANT FROM YC AND SO
260  REM  THAT (YH-YL)/(XH-XL)=S:
270 YH = YC + S * (XH - XL) / 2
280 YL = YC - S * (XH - XL) / 2
290 PI = 4 *  ATN (1): REM  COMPUTER'S PI
300  HOME
310  REM  SET THETA VALUES:
320 TL = 0: REM  LOW THETA
330 TH = 2 * PI: REM  HIGH THETA
340 BOUND = 70: REM  NUMBER OF POINTS TO PLOT
350 TS = (TH - TL) / BOUND: REM  THETA STEP SIZE
360  HGR : HCOLOR= 3
370  REM  DISPLAY ECC AND DIRECTRIX:
380  VTAB (21)
390  PRINT "DIRECTRIX X = ";P
400  PRINT "ECCENTRICITY = ";ECC
410  REM  PLOT ORIGIN:
420 X = 0:Y = 0
430  GOSUB 570: REM  PLOTTER SUBROUTINE
440  REM  DRAW DIRECTRIX:
450  IF P > XH OR P < XL THEN  PRINT "DIRECTRIX OFF SCREEN":
GOTO 490
460 U = 279 * (P - XL) / (XH - XL)
470  HPLOT U,0 TO U,159
480  REM  DRAW CONIC SECTION:
490  FOR T = TL TO TH STEP TS
```

```
500 R = ECC * P / (1 + ECC *  COS (T))
510  REM     CONVERT (R,THETA) TO (X,Y):
520 X = R *  COS (T)
530 Y = R *  SIN (T)
540  GOSUB 570: REM  PLOT X,Y
550  NEXT
560  GOTO 740: REM  ENDING OPTIONS
570  REM  PLOTTER SUBROUTINE
580  REM
590  REM  CONVERT X,Y TO U,V:
600  REM
610  REM   X=XL TO U=0
620  REM   X=XH TO U=279
630  REM
640  REM   Y=YL TO V=159
650  REM    Y=YH TO V=0
660  REM
670 U = 279 * (X - XL) / (XH - XL)
680 V =  - 159 * (Y - YH) / (YH - YL)
690  REM  IF U OR V OUT OF RANGE, SKIP PLOT:
700  IF U < 0 OR U > 279 THEN 730
710  IF V < 0 OR V > 159 THEN 730
720  HPLOT U,V
730  RETURN
740  REM  ENDING OPTIONS
750  VTAB 23
760  PRINT "DO ANOTHER (Y OR N)";
770  INPUT ANS$
780  IF ANS$ = "N" THEN  HGR : TEXT : HOME : END
790  PRINT "CLEAR SCREEN";: INPUT ANS$
800  IF ANS$ = "Y" THEN  HGR
810  PRINT "CHANGE ECCENTRICITY";: INPUT ANS$
820  IF ANS$ <  > "Y" THEN 840
830  PRINT "ECCENTRICITY =";: INPUT ECC
840  PRINT "CHANGE DIRECTRIX";: INPUT ANS$
850  IF ANS$ <  > "Y" THEN 870
860  PRINT "DIRECTRIX X = ";: INPUT P
870  PRINT "READY TO PLOT";: INPUT ANS$
880  IF ANS$ = "Y" THEN  HOME : GOTO 380
890  GOTO 760: REM  OPTIONS AGAIN
```

Lines 170 to 190 ask you for the eccentricity and directrix.

Lines 200 to 280 set the screen bounds.

Lines 310 to 350 set the bounds on θ for the plotting.

Lines 370 to 400 display the equation of the directrix and the value of the eccentricity below the graphics screen.

Lines 410 to 470 plot the focus and draw the directrix.

Lines 480 to 550 draw the conic, using equation (27-3) in line 500 and the conversion of R and θ to X and Y in lines 520 and 530. Line 540 calls the plotter subroutine (lines 570 to 730).

After the curve is drawn, lines 740 to 890 give you various options: Clear the screen, change the eccentricity, or change the directrix.

Exercise 27-13. Try out POLAR CONICS for various choices of eccentricity and directrix.

Exercise 27-14. A parabola has eccentricity = 1. To see what effect the location of the directrix has on the shape of the parabola, sketch, without clearing the screen, the parabolas with directrix given by

1. X = 2
2. X = 3
3. X = 4

Then clear the screen and sketch for directrix

4. X = −2
5. X = −3
6. X = −4

Exercise 27-15. Graph ellipses with eccentricity = 0.5 and directrix given by the equations

1. X = 2
2. X = 3
3. X = 4

Then clear the screen and sketch for directrix

4. X = −2
5. X = −3
6. X = −4

As the directrix gets farther away from the focus, does the ellipse get larger or smaller?

Exercise 27-16. Now for hyperbolas: Use eccentricity 1.5 and sketch the graph for directrix given by

1. X = 1
2. X = 2
3. X = 3
4. X = 4

In Exercises 27-13 through 27-15, we kept the eccentricity fixed and changed the directrix. Now let's do the opposite to see what effect the eccentricity has on the curve.

Exercise 27-17. For directrix X = 3, sketch the ellipse with eccentricity

1. ECC = 0.3
2. ECC = 0.4

3. ECC = 0.5
4. ECC = 0.6
5. ECC = 0.7
6. ECC = 0.8

Which is more nearly a circle—an ellipse with eccentricity 0.3, or one with eccentricity 0.8?

Exercise 27-18. For directrix X = 3, draw the hyperbola with eccentricity

1. ECC = 1.4
2. ECC = 1.6
3. ECC = 1.8
4. ECC = 2

So far, we have kept either the directrix or the eccentricity fixed. Now let's change both.

Exercise 27-19. Draw the conic section with the given eccentricity and directrix. First draw all these without clearing the screen. Then, to get a better picture of what's happening, draw them again. This time clear the screen between each sketch.

1. ECC = 0.5, directrix X = 2
2. ECC = 0.25, directrix X = 4
3. ECC = 0.2, directrix X = 5
4. ECC = 0.1, directrix X = 10

What happens to the shape of the conic as the directrix gets farther from the focus?

In Exercise 27-18, we didn't choose the eccentricity and directrix at random. We chose eccentricity ECC and directrix X = P so that ECC*P = 1, and then looked at curves as the directrix got farther and farther away from the focus. To make the end result even more apparent, see what the next exercise gives.

Exercise 27-20. Sketch the conic for the given eccentricity and directrix. Clear the screen between sketches.

1. ECC = 0.05, directrix X = 20
2. ECC = 0.01, directrix X = 100
3. ECC = 0.005, directrix X = 200
4. ECC = 0.001, directrix X = 1,000

The results of these last two exercises show why we say that a circle has eccentricity 0 and a directrix at infinity. Thus, we can make the circles fit the eccentricity-directrix format if we are willing to stretch the language a bit. But what about the equations? Can we make the circle equation fit the noncircle equation form? For a circle, we have the polar equation R = a. For a noncircle conic, we have equation (27-3):

$$R = \frac{ECC*P}{1 + ECC*\cos(\theta)}$$

Look at what happens to this equation if we let ECC approach 0 and P approach infinity in such a way that ECC*P is a constant, say ECC*P = a. Since cos(θ) is always between −1 and +1, the term ECC*cos(θ) in the denominator shrinks away to 0 because ECC approaches 0. Then the equation becomes

$$R = ECC*P$$

or

$$R = a$$

Thus, we can even make the circle equation show up as the result of taking eccentricity 0 and directrix at infinity!

Finally, try to use your mathematics to create some art work.

Exercise 27-21. Use POLAR GRAPHER and standard functions, such as T, sin(T), cos(T), tan(T), |T|, log(T), and so forth to create functions whose graphs are similar to those shown in Figure 27-5.

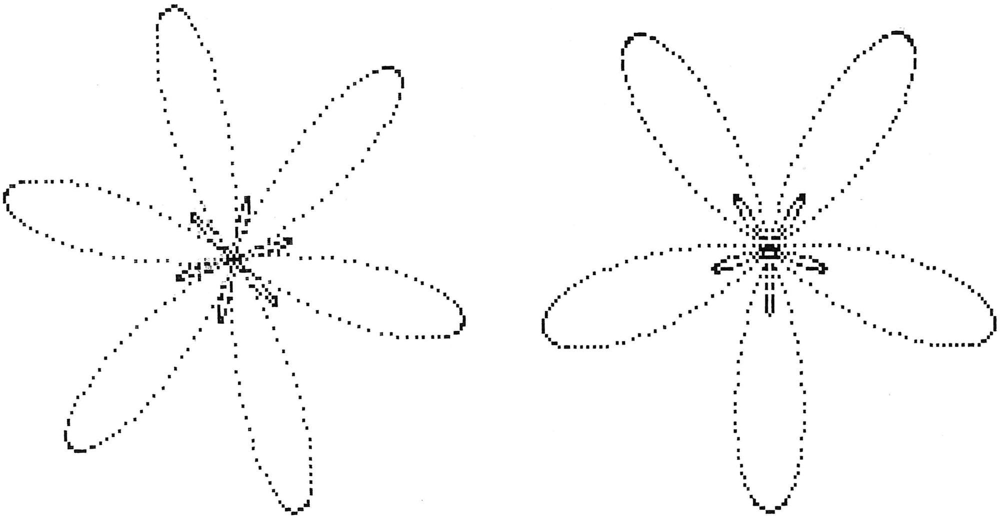


Figure 27-5 Polar graphs

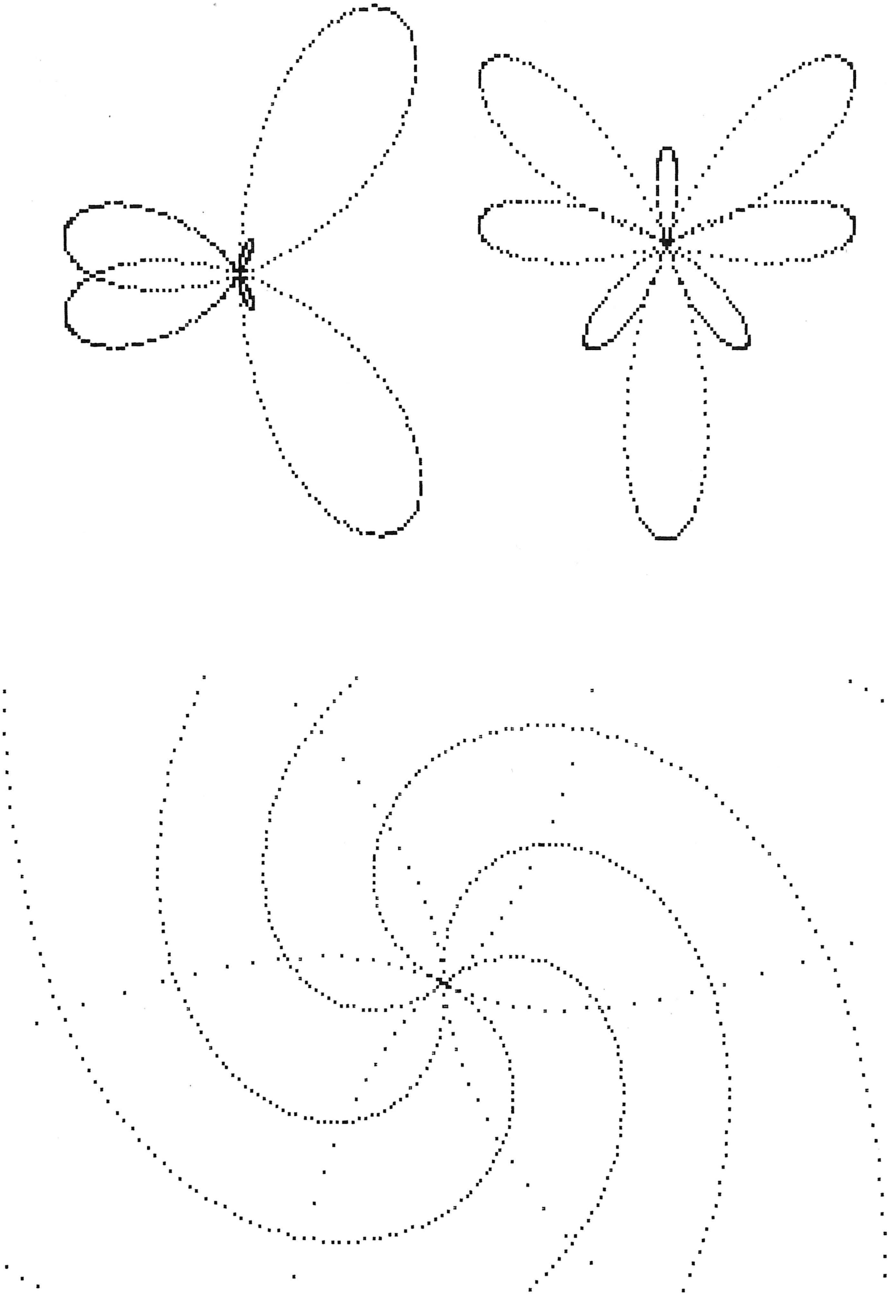


Figure 27-5 *(continued)*

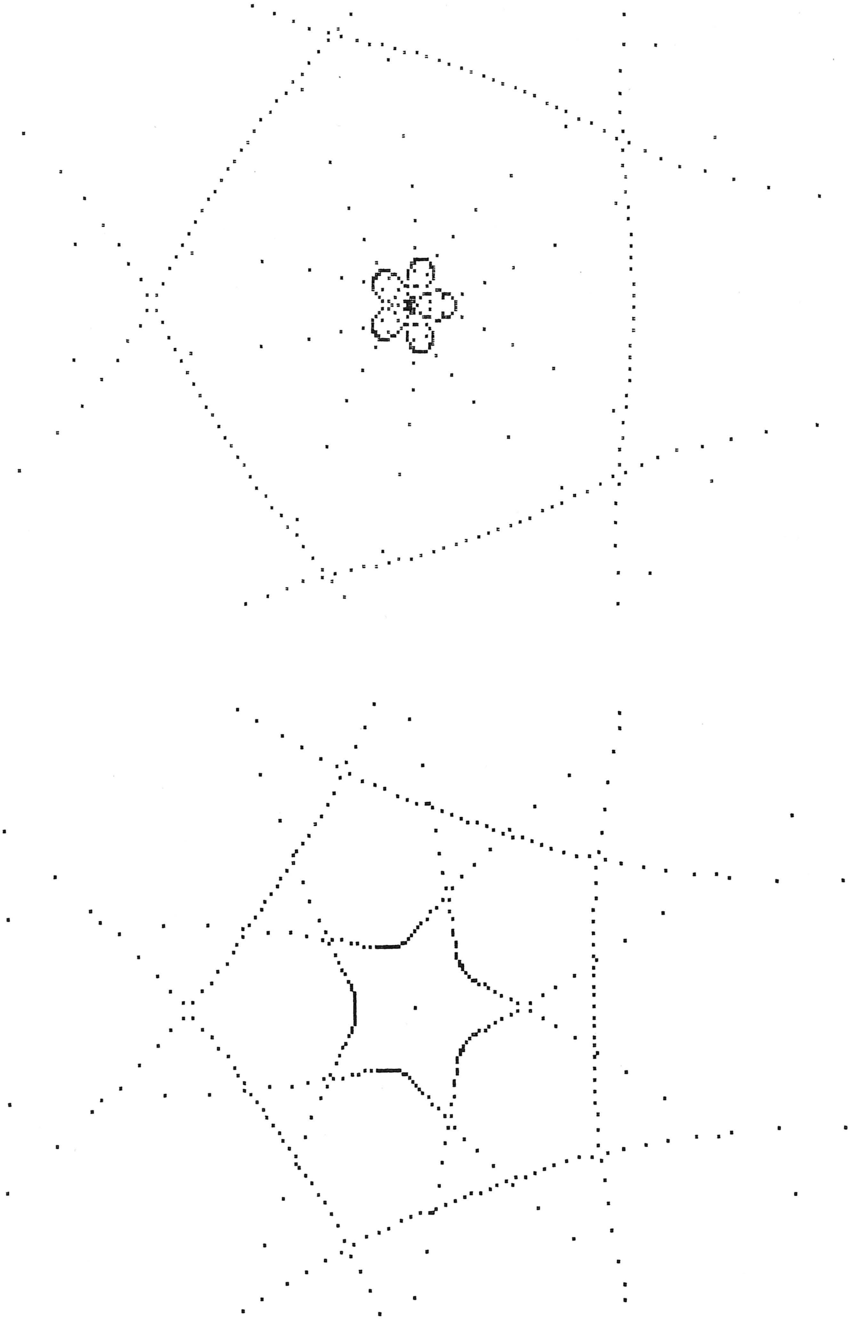


Figure 27-5 (*continued*)

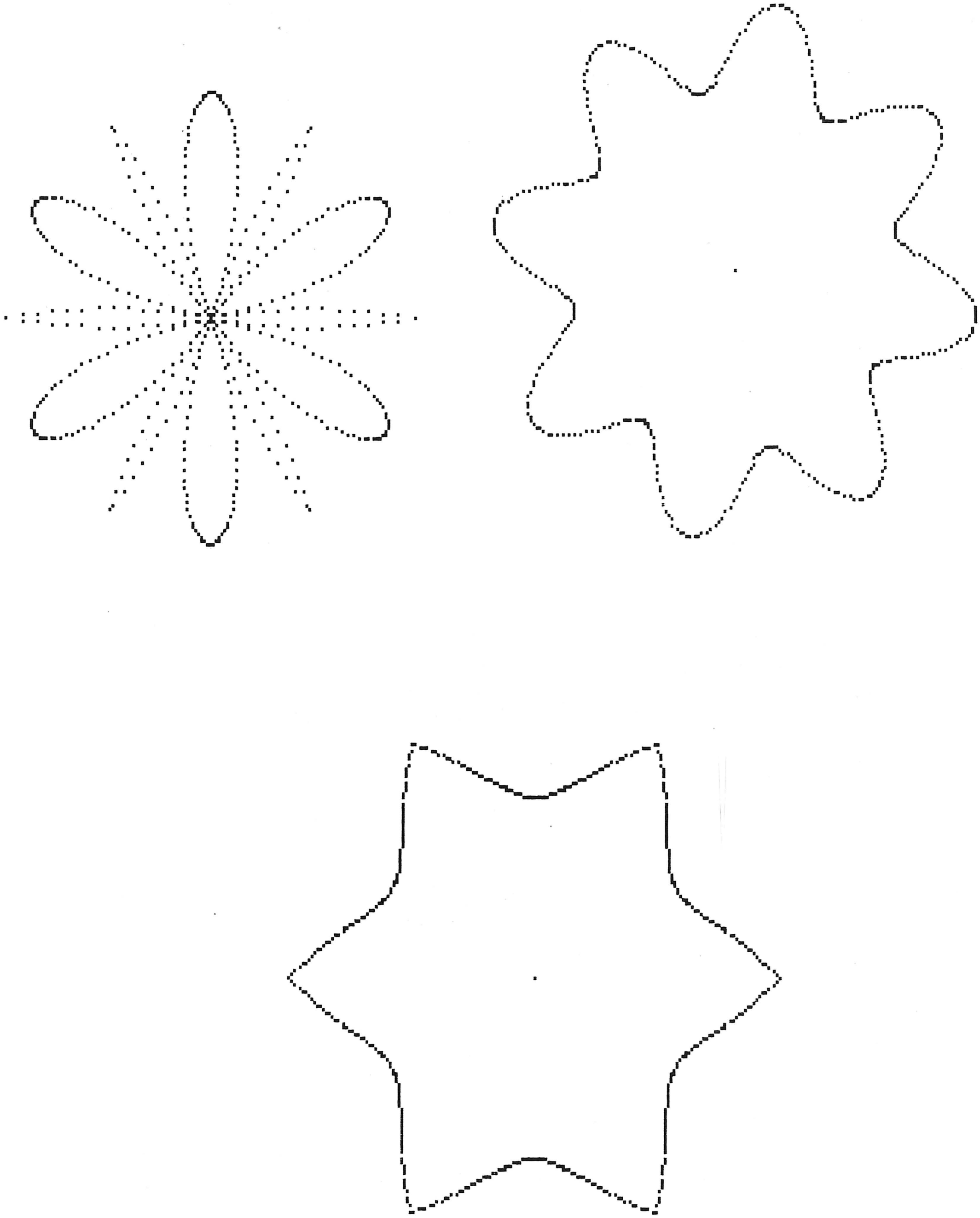
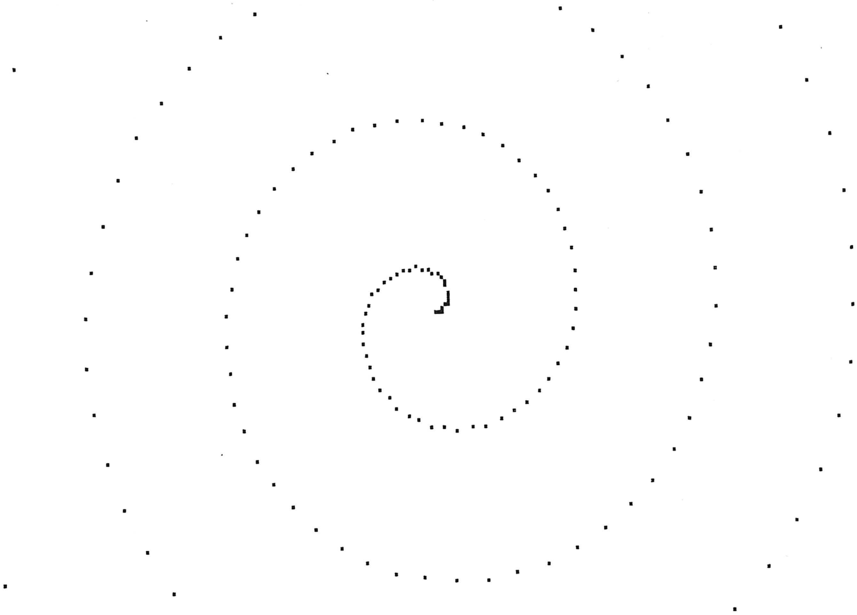


Figure 27-5 (*continued*)



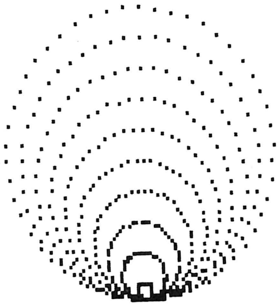


Figure 27-5 (*continued*)

28

LIMITS

The idea of the limit of a function is crucial to the development of calculus because the derivative is defined as a certain limit. In this chapter, we study the intuitive idea of limit.

To say that a function F(X) has a number L as *limit* as X approaches a base point X0 means that the value of F(X) can be made arbitrarily close to L if X is taken sufficiently close to (but different from) X0.

You can use the program LIMIT TESTER (see Listing 28-1) to test whether a function has a limit at a given point. This program computes the values of a function F(X) at a sequence of points X0 + DX for DX getting closer and closer to 0.

LISTING 28-1

```
100  REM    LIMIT TESTER
110  HOME
120  PRINT "    THIS PROGRAM COMPUTES THE VALUES"
130  PRINT : PRINT "OF A FUNCTION F(X) AS X APPROACHES A"
140  PRINT : PRINT "BASE POINT XO."
150  PRINT : PRINT "     YOU SPECIFY THE BASE POINT AND"
160  PRINT : PRINT "SAY WHETHER TO APPROACH XO FROM"
170  PRINT : PRINT "THE RIGHT OR LEFT."
180  PRINT : PRINT "     THE FUNCTION IS DEFINED IN LINE"
190  PRINT : PRINT "300:"
200  LIST 300
210  PRINT : PRINT "DO YOU WANT TO CHANGE THE FUNCTION";
220  INPUT ANS$
230  IF ANS$ <  > "Y" THEN 290
240  HOME
250  PRINT "ENDING SO YOU CAN CHANGE THE FUNCTION."
260  LIST 300
270  PRINT : PRINT "TO RESTART, TYPE 'RUN 300'
```

```
280  END
290  HOME
300  DEF  FN F(X) = X * X
310  PRINT "BASE POINT X0 = ";: INPUT X0
320  PRINT : PRINT "APPROACH FROM LEFT OR RIGHT (L OR R)";
330  INPUT DIR$
340  REM  INITIALIZE DX:
350  IF DIR$ = "L" THEN DX =  - 1
360  IF DIR$ = "R" THEN DX = 1
370  IF DIR$ <  > "L" AND DIR$ <  > "R" THEN 320
380  HOME
390  PRINT "HIT 'S' TO STOP, RETURN TO CONTINUE."
400  PRINT "BASE POINT X0 = ";X0
410  POKE 34,3: REM    PROTECT FROM SCROLLING
420  VTAB 4
430 X = X0 + DX: REM    NEW X
440 Y =  FN F(X): REM   NEW Y
450  PRINT "X = ";X;"  Y = ";Y
460 DX = DX * .9: REM    NEW DX
470  GET ANS$
480  IF ANS$ = "S" THEN 500: REM  END
490  GOTO 430: REM  DO ANOTHER ONE
500  POKE 34,0: REM  END SCROLLING PROTECTION
510  PRINT : PRINT "DO ANOTHER ONE WITH SAME FUNCTION?"
520  PRINT : PRINT "( Y OR N )";
530  INPUT ANS$
540  IF ANS$ <  > "N" THEN 290
```

Before trying out the program, let's analyze it.

Lines 100 to 290 print a description of the program, show the definition of the function F(X), and give you a chance to change the function.

Line 300 gives the definition of the function.

Line 310 sets the base point X0.

Lines 320 and 330 let you set the direction from which X will approach X0.

Lines 340 to 370 initialize the X step size DX according to your choice of direction. If you say to approach X0 from the right, DX is taken to be 1 so that X starts off at X0 + 1. If you say to approach from the left, X starts at X0 − 1.

Lines 390 and 400 print some instructions and the value of the base point on the screen.

Lines 430 and 440 set X = X0 + DX and Y = F(X).

Line 450 prints X and Y.

Line 460 gets the new X step size DX by multiplying the old value by 0.9.

Now try out LIMIT TESTER in Exercise 28-1.

Exercise 28-1. LIMIT TESTER comes with the function X^2, so test the limit for this function at each of the following base points X0. Try coming in both from the left and the right. Check that the X values do indeed approach the base point from the direction you specify. Also note that the values of F(X) approach $(X0)^2$ as you would expect.

1. X0 = 0
2. X0 = 1
3. X0 = −1
4. X0 = 2
5. X0 = −2

Exercise 28-2. Repeat Exercise 28-1 for the function $F(X) = X^3$.

In both Exercises 28-1 and 28-2 you knew from looking at the functions what limit to expect, namely the value F(X0) of the function at the base point X0. Such functions are called *continuous*. Specifically, a function F(X) is continuous if at each point X0 where F(X0) is defined, the limit of F(X) is F(X0).

The more important and interesting limit considerations occur for functions that are undefined at the base point X0. For example, the function

$$F(X) = \frac{X^2 - 1}{X - 1} \qquad (28\text{-}1)$$

is not defined at $X0 = 1$. Fortunately, the idea of a limit for F(X) at $X0 = 1$ still makes sense: Although the limit definition requires that we consider X points close to X0, it only requires that we test X points *different from* X0. Note that LIMIT TESTER never has $X = X0$ because it starts with $DX = -1$ or $+ 1$ and then repeatedly multiplies DX by 0.9. Hence, DX is never 0, so $X = X0 + DX$ is never equal to X0.

Exercise 28-3. Investigate the limit of function (28-1) at $X0 = 1$.

1. Approach X0 from the right and do several steps. What seems to be the limiting value of F(X)?
2. Repeat part 1, but this time approach X0 from the left.

For the function (28-1), we can use algebra to check that the computer gives us the correct limit: Factoring the numerator gives

$$F(X) = \frac{(X + 1)\,(X - 1)}{(X - 1)}$$

so, if $X \neq 1$,

$$F(X) = X + 1$$

Thus we can see that the limit of F(X) at $X0 = 1$ is $1 + 1 = 2$. Is that the limit you found in Exercise 28-3?

Now that we know the limit of this function is 2, let's try the computer on it again.

Exercise 28-4. Use LIMIT TESTER once again on function (28-1) at the point $X0 = 1$.

1. Come in from the right and do more and more steps (you may want to hold down the REPT and RETURN keys to speed things up). Watch what happens to X and F(X).
2. Repeat part 1; this time come in from the left.

The results of Exercise 28-4 show that you can push the computer so hard doing limits that the output gets strange. For the first several steps, F(X) gets closer and closer to

2, but eventually F(X) wanders around. The numbers you get depend on just how you typed in the formula for the function, but I got values of 2.5, 5, and 1.5 near the end. The values for X also show limitations of the machine. Eventually the computer reports X = 1 and gives some numerical value for Y. Well, we know that the true value of X cannot be 1, because X = X0 + DX and DX is never really zero. Moreover, if X were really 1, the denominator in the function would be 0, so Y would be undefined. In fact, you do eventually get a division-by-zero error, so the computer finally can't distinguish between X and 1 anymore.

So you see that you have to be somewhat careful in using the computer to investigate limits. If X is too close to the base point X0, the computer may be unable to give meaningful output. On the other hand, it can still give a lot of useful information about limits.

Exercise 28-5. Investigate the limit of

$$F(X) = \frac{(X^3 - 1)}{(X - 1)}$$

at X0 = 1. Does the function seem to approach the same value as X approaches X0 from the right as from the left?

Exercise 28-6. Investigate the limit of

$$F(X) = \frac{|X|}{X}$$

at X0 = 0. What is the limiting value of F(X) as X approaches X0

1. from the right?
2. from the left?

In Exercise 28-6, you get a limit of 1 as X approaches 0 from the right, but a limit of −1 as X approaches 0 from the left. For this reason, we say that the *left-hand limit* of F(X) is −1 and the *right-hand limit* is 1. Our original discussion of limit did not mention these directions separately, because there we were dicussing a *two-sided* limit. In the two-sided case, F(X) must get close to the limit L for X on *both* sides of X0. Since the function |X|/X fails to approach the *same* value from both sides, it does not have a two-sided limit at X0 = 0.

Exercise 28-7. For each of the following functions, find its left-hand and right hand limits at the base point, and determine whether it has a two-sided limit at the point.

1. $$F(X) = \frac{|X - 3|}{X - 3}, \quad X0 = 3$$

2. $$F(X) = X + 1 + \frac{|X + 3|}{X + 3}, \quad X0 = -3$$

3. $$F(X) = \frac{X^2 - 9}{X + 3}, \quad X0 = -3$$

4. $$F(X) = \frac{X^3 + 8}{X + 2}, \quad X0 = -2$$

So far, the functions have had finite one-sided limits at the base point. The next exercise shows a different kind of behavior.

Exercise 28-8. Investigate the right-hand and left-hand limits of

$$F(X) = \frac{X^2 + 1}{X - 1}$$

at X0 = 1.

This function blows up as X approaches 1. For X coming in toward 1 from the right, F(X) gets larger and larger without bound. So, we say that F(X) has a right-hand limit $+\infty$ at X0 = 1. As X approaches 1 from the left, F(X) is negative and its absolute value grows without bound. In this case, we say the left-hand limit is $-\infty$. Exercise 28-9 describes another function with infinite one-sided limits.

Exercise 28-9. Investigate the one-sided limits of

$$F(X) = \frac{X^3 + 3}{X + 2}$$

at X = −2.

We have now seen two types of limiting behavior—finite and infinite one-sided limits. The next exercise shows that there are even more possibilities.

Exercise 28-10. Investigate the one-sided limits of

$$F(X) = \sin\left(\frac{1}{X}\right)$$

at X0 = 0.

This function always takes on values between −1 and + 1, so it certainly doesn't have an infinite limit at 0. On the other hand, it refuses to settle down to any specific value. As X approaches 0 from the right, 1/X gets larger and larger, and sin(1/X) keeps running through all the values from −1 to +1. Thus, this function does not have one-sided limits at X0 = 0. The next exercise gives you a chance to study all the varieties of limits we've been discussing.

Exercise 28-11. For each of the following functions, determine its left-hand and right-hand limits, if any, at the base point. Note whether it has an infinite limit or a two-sided limit.

1. $$F(X) = \frac{X}{1 - X}, \quad X0 = 1$$

2. $$F(X) = \frac{X}{1 - X}, \quad X0 = 3$$

3. $$F(X) = \frac{1 + X}{1 + X^5}, \quad X0 = -1$$

4. $$F(X) = \frac{\sqrt{X} - 2}{X - 4}, \quad X0 = 4$$

5. $$F(X) \quad \frac{\sqrt{X} - 3}{X - 9}, \quad X0 = 9$$

6. $$F(X) = 4 + 3\,\frac{|X - 2|}{X - 2}, \quad X0 = 2$$

7. $$F(X) = \cos\left(\frac{1}{X}\right), \quad X0 = 0$$

8. $$F(X) = 2\arctan\left(\frac{1}{X}\right), \quad X0 = 0$$

In Exercise 28-4, you saw that the computer can be pushed too far in attempting to evaluate a limit. If X and X0 are too close together, the computer is not able to distinguish between them, and nonsense may result. To further illustrate this, consider the function in the following exercise.

Exercise 28-12. Investigate the one-sided limits of

$$F(X) = 4 + \frac{1}{1{,}000X}$$

at X0 = 0.

Because the denominator of the second term is zero at X = 0, whereas the numerator is not, the second term will blow up at X = 0. However, since the X in the denominator is multiplied by 1,000, the effect of the second term is obscured until X gets rather close to 0. Hence, it may take several steps of LIMIT TESTER to see that the one-sided limits are infinite. By way of contrast, the function in part 1 of Exercise 28-11 reveals its infinite behavior much sooner.

For something even worse in this limit-obscuring line, think about the function

$$F(X) = 4 + \frac{1}{10^{1000}X}$$

We can see that the function has infinite one-sided limits at X0 = 0 because of the second term. However, because the X in the denominator is multiplied by 10^{1000}, X has to be *very* close to 0 for this behavior to show up, so close, in fact, that most computers wouldn't even be able to show it. (Try LIMIT TESTER on it and see!) This example shows that our knowledge of the formula of a function is often more effective in determining limits than our use of the computer.

Now that we have studied limits of functions at finite points X0, let's turn to limits at infinity. To say that a function F(X) has limit L as X goes to $+\infty$ means that F(X) can be made arbitrarily close to L if X is just taken large enough. To investigate such limits with the computer, we must compute F(X) at ever-increasing X values. This can be done easily by modifying LIMIT TESTER. There, when coming in from the right, F(X) is computed at X = X0 + DX, where DX starts at 1 and DX is made smaller at each step by multiplying it by 0.9. If, instead, we multiply DX by 1.1 at each step, X = X0 + DX marches off to $+\infty$, just as we want.

Exercise 28-13. Change line 460 of LIMIT TESTER to

```
460 DX = DX*1.1
```

What about the limit of F(X) as X goes to $-\infty$? Now we want X to move to the left without bound. The modification you made in Exercise 28-13 takes care of this case too: For going to the left, DX is set to −1 in line 350, then repeatedly multiplying DX by 1.1 forces X = X0 + DX to move off to the left.

In taking a limit at $+\infty$ or $-\infty$, the point X0 just gives some place to start computing. Hence, you may wish to change line 310 so that it prints ''STARTING POINT X0 =''. To make the modified program's request for a direction correspond better to this new situation, you may wish to change the question in line 320 to something like ''MOVE X TO LEFT OR RIGHT?''.

Exercise 28-14. Investigate the limits of each of the following functions as X approaches $+\infty$ and as X approaches $-\infty$.

1. $F(X) = X^2$
2. $F(X) = X^3$
3. $F(X) = X^4$

Each of the functions in Exercise 28-14 gives an infinite limit in the two directions. Exercise 28-15 deals with some functions that give finite limits.

Exercise 28-15. Investigate the limits of each of the following functions as X approaches $+\infty$ and $-\infty$. In each case, write down the limit you find.

1. $$F(X) = \frac{2X - 1}{3X + 1}$$

2. $$F(X) = \frac{4X + 2}{2X - 19}$$

3. $$F(X) = \frac{3X^2 - X + 1}{4X^2 + X - 3}$$

$$F(X) = \frac{X^3 + 10X^2 + 7}{4X^3 - 2X - 3}$$

Each of the functions above is a rational function, that is, a quotient of polynomials. In each case, the degrees of the numerator and denominator are equal, and the limits are finite numbers.

Exercise 28-16. Based on your results in Exercise 28-15, find a formula that gives the limiting value of such a rational function in terms of the coefficients of the polynominals in the numerator and denominator of the quotient.

Now let's look at some rational functions where the degrees of numerator and denominator are not equal.

Exercise 28-17. Investigate the limits at $+\infty$ and $-\infty$ for each of the following functions:

1. $$F(X) = \frac{1}{X - 5}$$

2. $$F(X) = \frac{X}{X^2 - X + 3}$$

3. $$F(X) = \frac{X^2 + X - 2}{X + 100}$$

4. $$F(X) = \frac{X^3 + 1}{100X^2 + 30X + 5}$$

Exercise 28-18. Each of the functions in Exercise 28-17 has limits of 0 or $\pm\infty$. Based on your results of that exercise, formulate a procedure or determine, just by looking at such a rational function, whether it goes to 0 or to $\pm\infty$.

In general, when you take the limit of a quotient N(X)/D(X) as X approaches $+\infty$ or $-\infty$, you're having a contest between the numerator N(X) and the denominator D(X) to see which goes toward infinity more rapidly. If the limit is 0, then D(X) wins the race. If the limit is infinite, then N(X) wins. Thus, you can use this limit investigation to compare the growth rate of functions.

Exercise 28-19. Investigate the limit of each of the following quotients at $+\infty$ and state which function grows faster, the numerator or the denominator.

1. $$\frac{2^x}{X^{10}}$$

2. $$\frac{\sin(X)}{X}$$

3. $$\frac{e^{x}}{X^{10}}$$

4. $$\frac{[\log(X)]^{10}}{X}$$

Exercise 28-20 gives one final example of limits at infinity:

Exercise 28-20. Find the limit of $(1 + 1/X)^x$ as X goes to $+\infty$. Do you recognize the limit?

29

DERIVATIVES

In Chapter 28, "Limits," we studied the idea of the limit of a function at a point. Since the derivative is defined as a limit, we can use the computer to investigate derivatives as well.

The *derivative* of a function F(X) at a point X0 is defined as the limit of the quotient

$$\frac{[F(X0 + DX) - F(X0)]}{DX} \qquad (29\text{-}1)$$

as DX approaches 0.

Geometrically, the derivative of F(X) at X0 gives the slope of the tangent line to the graph of F(X) at the point (X0, F(X0)). To see this, consider the graph of the function in Figure 29-1. Moving away from X0 DX units along the X axis gives a point X0 + DX on the axis and hence a point [X0 + DX, F(X0 + DX)] on the curve. The two points on the curve determine a line, called a *secant* line, which should be close to the tangent line if DX is small. The slope of this secant line is precisely the quotient (29-1). Thus the derivative is the limit of the slopes of the secant lines and hence gives the slope of the tangent line.

For a given function, we can compute quotient (29-1) for values of DX approaching 0 to see what the derivative is. The program DERIVATIVE LIMIT (see Listing 29-1) is set up to do exactly that.

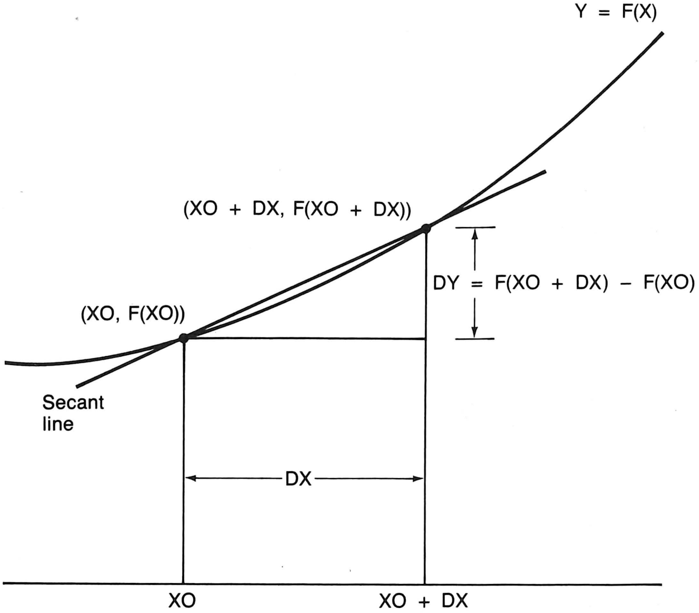


Figure 29-1 Secant line

LISTING 29-1

```
100  REM    DERIVATIVE LIMIT
110  HOME
120  PRINT "    THIS PROGRAM APPROXIMATES THE"
130  PRINT : PRINT "DERIVATIVE OF A FUNCTION F(X) AT"
140  PRINT : PRINT "A POINT XO."
150  PRINT : PRINT "    IT COMPUTES THE QUOTIENT"
160  PRINT : PRINT "    DY/DX = (F(X)-F(XO))/DX"
170  PRINT : PRINT "FOR X = XO + DX APPROACHING XO."
180  PRINT : PRINT "     YOU SPECIFY THE BASE POINT AND"
190  PRINT : PRINT "SAY WHETHER TO APPROACH XO FROM"
200  PRINT : PRINT "THE RIGHT OR LEFT."
210  VTAB 23
220  PRINT "PRESS ANY KEY TO CONTINUE."
230  GET ANS$
240  HOME
250  PRINT : PRINT "     THE FUNCTION IS DEFINED IN LINE"
260  PRINT : PRINT "370:"
270  LIST 370
280  PRINT : PRINT "DO YOU WANT TO CHANGE THE FUNCTION";
290  INPUT ANS$
300  IF ANS$ <  > "Y" THEN 360
310  HOME
320  PRINT "ENDING SO YOU CAN CHANGE THE FUNCTION."
330  LIST 370
340  PRINT : PRINT "TO RESTART, TYPE 'RUN 360'
350  END
360  HOME
370  DEF  FN F(X) = X ^ 2
380  PRINT "BASE POINT XO = ";: INPUT XO
390  PRINT : PRINT "APPROACH FROM LEFT OR RIGHT (L OR R)";
```

```
400   INPUT DIR$
410   REM  INITIALIZE DX:
420   IF DIR$ = "L" THEN DX =  - 1
430   IF DIR$ = "R" THEN DX = 1
440   IF DIR$ <  > "L" AND DIR$ <  > "R" THEN 390
450   HOME
460   PRINT "HIT 'S' TO STOP, RETURN TO CONTINUE."
470   PRINT "BASE POINT XO = ";XO
480   POKE 34,3: REM    PROTECT FROM SCROLLING
490   VTAB 4
500 X = XO + DX: REM   NEW X
510 DY =  FN F(X) -  FN F(XO): REM    NEW DY
520   PRINT "X = ";X;"  DY/DX = ";DY / DX
530 DX = DX * .5: REM    NEW DX
540   GET ANS$
550   IF ANS$ = "S" THEN 570: REM  END
560   GOTO 500: REM  DO ANOTHER ONE
570   POKE 34,0: REM  END SCROLLING PROTECTION
580   PRINT : PRINT "DO AGAIN WITH THIS FUNCTION (Y OR N)";
590   INPUT ANS$
600   IF ANS$ <  > "N" THEN 360
610   END
```

This program is quite similar to the program LIMIT TESTER of Chapter 28.

Lines 100 to 350 print a program description and give you a chance to change the function.

Line 370 defines the function F(X).

Line 380 sets the base point X0.

Lines 390 to 440 initialize DX to +1 or −1 depending on whether you want to approach X0 from the right or left.

Lines 450 to 480 print the base point and stopping instructions on the screen.

Line 500 computes the new X value.

Line 510 computes the DY value.

Line 520 prints X and DY/DX.

Line 530 cuts DX in half to move the next point closer to X0.

Lines 540 to 560 pause while you look at the results and decide whether to compute for another DX value. If you decide not to, lines 570 to 610 give you the chance to change to another base point.

Exercise 29-1. Run DERIVATIVE LIMIT for the function X^2 it comes with.

1. Take X0 = 3 and come in from the right. At the first step, X is 4. For that point, what is the slope of the secant line? What is the slope of the secant line for the next point?
2. Take X0 = 3 again, but come in from the left this time. At the first step, X is 2. For that point, what is the slope of the secant line? What is the slope of the secant line for the next point?
3. Again take X0 = 3 and come in from the right. Do enough steps that the first three decimal places of DX stabilize. What seems to be the limit of DY/DX?
4. Again with X0 = 3, this time come in from the left. Do enough steps that the first

three decimal places of DY/DX stabilize. What seems to be the slope of the tangent line at X0 = 3?

Exercise 29-2. Repeat Exercise 29-1 for F(X) = X^2 and the following base points:

1. X0 = −3
2. X0 = 2
3. X0 = −2
4. X0 = 0

Exercise 29-3. For the function F(X) = $\sqrt{X}$, use DERIVATIVE LIMIT to guess at the derivative of F(X) at each of the following points. Come in from both the right and the left and check that you get the same limit both ways.

1. X = 1
2. X = 4
3. X = 9
4. X = 16

Exercise 29-4. Investigate the derivative of F(X) = |X| at X0 = 0.

1. What right-hand limit does DY/DX have?
2. What left-hand limit does it have?

The fact that the left- and right-hand limits of DY/DX are different for F(X) = |X| at X0 = 0 shows that this function does not have a derivative at X0 = 0.

It is easy to see from the graph of F(X) = |X| that it does not have a derivative at X0 = 0, because the graph has a sharp corner at (0, 0). So, the graph does not have a single, well-defined slope there. (If you don't already know what the graph of |X| looks like, use your graphing program from Exercise 22-4 to draw it.)

Exercise 29-5. What about the function F(X) = X*|X|? Does it have a derivative at X0 = 0? See what you can tell by using DERIVATIVE LIMIT.

The true power of the derivative comes from thinking of it as a new function that can give information about the original function F(X). At each point X0 where the limit of DY/DX exists, we have a number F′(X0) equal to the limit of DY/DX at that point. This correspondence of X0 to F′(X0) gives the *derivative function* F′(X).

It is easy to get the computer to approximate the derivative function. We just have to change the program DERIVATIVE LIMIT a bit. In that program, we kept the base point X0 fixed and changed DX so we could observe the limit process. Now we'll keep DX fixed and change X0 so we can see the function F′(X0). The value we give DX must be small, say DX = 0.001, so DY/DX will give (we hope) a good approximation to the derivative. The program DERIVATIVE FUNCTION (see Listing 29-2) uses this new arrangement.

LISTING 29-2

```
100  REM     DERIVATIVE FUNCTION
110  HOME
120  PRINT "    THIS PROGRAM APPROXIMATES THE"
130  PRINT : PRINT "DERIVATIVE OF A FUNCTION F(X)."
140  PRINT : PRINT "    IT COMPUTES THE QUOTIENT"
150  PRINT : PRINT : PRINT "    DY/DX = (F(X)-F(X0))/DX"
160  PRINT : PRINT : PRINT "FOR POINTS X0 OF YOUR CHOOSING."
170  PRINT : PRINT "    YOU SPECIFY IN ADVANCE THE X-SHIFT"
180  PRINT : PRINT "DX TO BE USED."
190  VTAB 23
200  PRINT "PRESS ANY KEY TO CONTINUE."
210  GET ANS$
220  HOME
230  PRINT : PRINT "    THE FUNCTION IS DEFINED IN LINE"
240  PRINT : PRINT "350:"
250  LIST 350
260  PRINT : PRINT "DO YOU WANT TO CHANGE THE FUNCTION";
270  INPUT ANS$
280  IF ANS$ <  > "Y" THEN 340
290  HOME
300  PRINT "ENDING SO YOU CAN CHANGE THE FUNCTION."
310  LIST 350
320  PRINT : PRINT "TO RESTART, TYPE 'RUN 340'
330  END
340  HOME
350  DEF  FN F(X) = X ^ 2
360  PRINT : PRINT "DX = ";: INPUT DX
370  HOME
380  REM  DISPLAY FUNCTION:
390  LIST 350
400  REM   WIPE OFF '350 DEF FN' :
410  VTAB 2
420  PRINT "                 "
430  REM  PRINT ENDING INSTRUCTION:
440  VTAB 4
450  PRINT "TYPE 'S' TO STOP."
460  REM  PROTECT FROM SCROLLING:
470  POKE 34,5
480  VTAB 6
490  INPUT "X0 = ";X0$
500  REM  EVALUATE X0 STRING:
510 X0 =  VAL (X0$)
520  REM  SEE IF X0$ WAS A NUMBER:
530  IF X0 <  > 0 THEN 570: REM  X0$ WAS A NUMBER
540  REM   HERE, X0=0, SO SEE IF X0$ WAS '0'
550  REM   IF NOT, ASSUME X0$ WAS 'S'
560  IF X0$ <  > "0" THEN 620: REM  END
570  REM  COMPUTE DY/DX:
580 X = X0 + DX
590 DY =  FN F(X) -  FN F(X0)
600  PRINT "           DY/DX = ";DY / DX
610  GOTO 490: REM  DO ANOTHER ONE
620  POKE 34,0: REM  END SCROLLING PROTECTION
630  END
```

Lines 100 to 330 print a description of the program and give you a chance to change the function.

Line 350 defines the function F(X).

Line 360 gets the value for DX.

Lines 370 to 470 print the function definition and ending instructions on the screen and then protect those lines from scrolling.

Line 490 starts the main part of the program. So that you can stop the program by typing in ''S'', the program inputs the point X0 as a string, then checks whether the string was a number. It does this by seting X0 = VAL(X0$). If X0$ is not a number, VAL makes X0 = 0. Thus, if X0 has value 0 but X0$ is not the string ''0,'' the entered string was not a number and the program ends.

If the string was a number, the program computes the nearby X,

$$X = X0 + DX$$

in line 580, and the corresponding value of DY,

$$DY = F(X) - F(X0)$$

in line 590. Then it computes and prints the quotient DY/DX in line 600.

Line 610 goes back for another X0 value.

Now put DERIVATIVE FUNCTION to use in the next exercise.

Exercise 29-6. For the function $F(X) = X^2$, compute the approximation to F′(X0) at each of the following points:

1. X0 = −3
2. X0 = −2
3. X0 = −1
4. X0 = 0
5. X0 = 1
6. X0 = 2
7. X0 = 3

Do you recognize the correspondence here? In each case, the approximate value of F′(X0) given by DY/DX is almost equal to 2(X0). This suggests that the derivative function for $F(X) = X^2$ is $F'(X) = 2X$.

Exercise 29-7. For the function F(X) = 7X, approximate the derivative at several points. What seems to be the derivative function?

Exercises 29-6 and 29-7 show that you might be able to use this program to discover derivative formulas. The next exercise gives you some more to try.

Exercise 29-8. For each of the following functions, approximate the derivative at several points. See if you can tell what the derivative function is.

1. $F(X) = X^3$
2. $F(X) = \log(X)$
3. $F(X) = X|X|$

As we showed in Chapter 28, you can't really be sure you know the limit of a function just by computing it at a few points. Here we're computing DY/DX at only one point and then pretending we know its limit. To really be sure you have guessed the correct derivative formula, you would have to deal directly with the function, in equation form, and show that the quotient DY/DX gives the derivative you guessed. Let's look at the function sin(X) to see an example of this.

The derivative of sin(X) at a point X0 is the limit of the quotient

$$\frac{DY}{DX} = \frac{\sin(X0 + DX) - \sin(X0)}{DX}$$

as DX approaches 0. To evaluate the limit, we must simplify the quotient. We can do this with the identity

$$\sin(X0 + DX) = \sin(X0)\cos(DX) + \cos(X0)\sin(DX)$$

Substituting this in DY gives

$$\begin{aligned} DY &= \sin(X0 + DX) - \sin(X0) \\ &= \sin(X0)\cos(DX) + \cos(X0)\sin(DX) - \sin(X0) \end{aligned}$$

Reordering the terms gives

$$DY = \sin(X0)\cos(DX) - \sin(X0) + \cos(X0)\sin(DX)$$

Factoring sin(X0) out of the first two terms gives

$$DY = \sin(X0)[\cos(DX) - 1] + \cos(X0)\sin(DX)$$

Then DY/DX becomes

$$\frac{DY}{DX} = \sin(X0)\,\frac{\cos(DX) - 1}{DX} + \cos(X0)\,\frac{\sin(DX)}{DX} \tag{29-2}$$

On the right-hand side, X0 is our fixed base point so sin(X0) and cos(X0) are fixed numbers. DX is the step size that we let approach zero. So, to know the limit, we need to know only the limit of each of the quotients

$$\frac{\cos(DX) - 1}{DX} \tag{29-3}$$

and

$$\frac{\sin(DX)}{DX} \tag{29-4}$$

as DX approaches 0.

To summarize: We used the identity for the sum of two angles to simplify DY/DX. Now we need only evaluate the limits of the quotients (29-3) and (29-4) to have the derivative of sin(X) at *any* point X0.

Exercise 29-9. Use the program LIMIT TESTER to see what the limits of the quotients (29-3) and (29-4) seem to be. That program works with a funtion F(X), and base point X0, so you will have to use

1. F(X) = (COS(X) − 1)/X, X0 = 0
2. F(X) = SIN(X)/X, X0 = 0

From your results in Exercise 29-9 it should seem that the limit of quotient (29-3) is 0 and that of quotient (29-4) is 1. With these limiting values, you see from Equation (29-2) that the limit of DY/DX is

$$\sin(X0)*0 + \cos(X0)*1 = \cos(X0)$$

This says that the derivative of sin(X) at X0 is cos(X0). You can use DERIVATIVE FUNCTION to test out this derivative formula for sin(X) in Exercise 29-10.

Exercise 29-10. Insert a line in DERIVATIVE FUNCTION to print the value of cos(X0) for each X0:

```
605 PRINT"     COS(";X0;") = ";COS(X0)
```

Run this version of the program and compare the approximate derivative to cos(X0) for various points X0.

An argument similar to the one we gave above for the derivative of sin(X) shows that the derivative of cos(X) is −sin(X). Test out this formula in Exercise 29-11.

Exercise 29-11. Compare the approximate derivative of cos(X) to −sin(X) at various points X0.

You saw in Exercise 29-8, part 2, that the derivative of log(X) seems to be 1/X. Since e^x and log(X) are inverse functions, we can use this derivative fact for log(X) to see what the derivative of e^x must be. The inverse function relationship gives

$$\log(e^x) = X$$

Differentiating both sides gives

$$\frac{1}{e^x}(e^x)' = 1$$

where we used the chain rule on the left-hand function. Solving this equation for $(e^x)'$ gives

$$(e^x)' = e^x$$

Now test this formula in Exercise 29-12.

Exercise 29-12. For $F(X) = e^x$, compare the approximate value of the derivative with e^{xo} at several points X0.

30

INTEGRALS

In Chapter 29, we investigated the derivative. Now we turn to the other main idea of calculus—the integral. Just as the derivative of a function F(X) has a geometric interpretation as the slope of a tangent line, the integral too has a geometric interpretation.

To start off, we'll assume the function F(X) is positive over an interval of X values, $A \le X \le B$. Then the *integral* of F(X) over this interval, which we write as

$$\int_A^B F(X)DX$$

is the area bounded by the graph of F(X), the X axis, and the lines X = A, X = B (See Figure 30-1.) If we restricted ourselves to straight-line functions F(X), the integral would be the area of a trapezoid, so we could easily compute it. For a general function, however, the upper boundary of the region is curved, so we have to approximate the area by some simple shapes such as rectangles.

Let's divide the interval $A \le X \le B$ into N equal pieces. Then each piece has length DX = (B − A)/N. We'll use N rectangles to approximate the area under the curve. (See Figure 30-2.) Each rectangle has its base of length DX on the X axis and its height determined by value of the function F(X) at the right-hand end point of the base. The right-hand end points are

$$A + DX,\ A + 2{*}DX,\ A + 3{*}DX,\ .\ .\ .,\ A + N{*}DX$$

so the heights of the rectangles are as follows:

Height of first rectangle: F(A + DX)

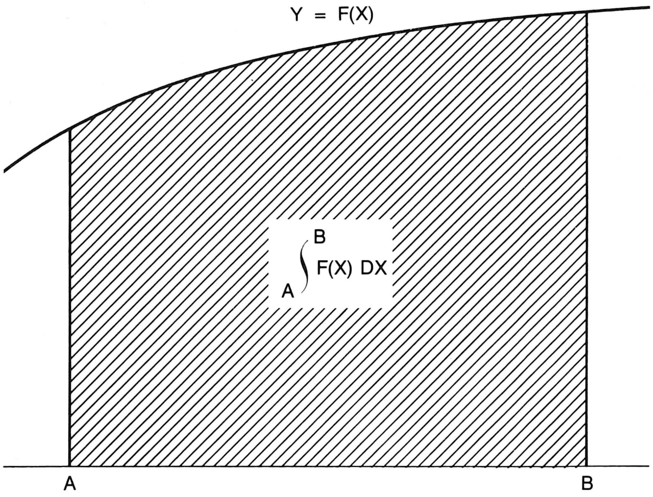


Figure 30-1 Area under curve

Height of second rectangle: F(A + 2*DX)
Height of third rectangle: F(A + 3*DX)
.
.
.
Height of last rectangle: F(A + N*DX)

Since the base of each rectangle has length DX, we get the area of each rectangle by multiplying its height by DX. Hence, the total area A(N) of the N rectangles is

$$A(N) = F(A + DX)*DX + F(A + 2*DX)*DX + F(A + 3*DX)*DX + \dots + F(A + N*DX)*DX$$

If we factor out the DX from the terms on the right-hand side, we get

$$A(N) = [F(A + DX) + F(A + 2*DX) + F(A + 3*DX) + \dots + F(A + N*DX)]*DX$$

To compute the area of the N rectangles, we must compute the sum of the function values F(A + K*DX), for K = 1 to N, and then multiply this sum by DX.

The area A(N) of the rectangles just gives an approximation to the integral, but it should be a fairly good approximation if N is large. Then each rectangle will be narrow, and there shouldn't be much difference between the straight rectangle tops and the curve. Thus, the integral is the limit of the rectangle areas A(N) as N goes to $+\infty$.

The method of approximating an integral by rectangles is called the *rectangle rule*. The program INTEGRAL LIMIT (see Listing 30-1) approximates the integral of a function by using the rectangle rule for ever-increasing N.

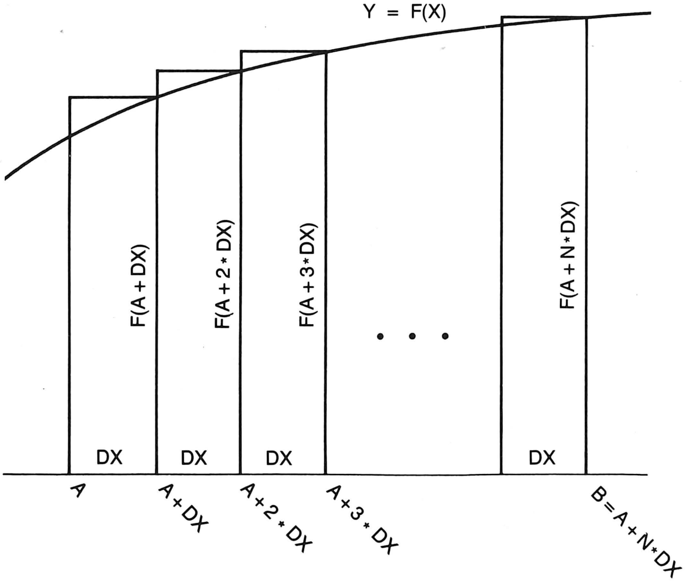


Figure 30-2 Rectangles approximating area

LISTING 30-1

```
100  REM      INTEGRAL LIMIT
110  HOME
120  PRINT "    THIS PROGRAM USES THE RECTANGLE"
130  PRINT : PRINT "RULE TO APPROXIMATE THE INTEGRAL"
140  PRINT : PRINT "OF A FUNCTION Y = F(X) FROM"
150  PRINT : PRINT "X = A TO X = B."
160  PRINT : PRINT
170  PRINT : PRINT "THE FUNCTION IS SPECIFIED IN LINE 290:"
180  LIST 290
190  PRINT : PRINT : PRINT "DO YOU WISH TO CHANGE THE FUNCTION
?"
200  PRINT : PRINT "(Y OR N)";: INPUT ANS$
210  IF ANS$ <  > "Y" THEN 270
220  HOME
230  PRINT : PRINT "ENDING SO YOU CAN CHANGE THE FUNCTION:"
240  LIST 290
250  PRINT : PRINT "TO RESTART, TYPE 'RUN 270'"
260  END
270  HOME
280  REM  DEFINE FUNCTION:
290  DEF  FN F(X) = X
300  PRINT : PRINT "STARTING POINT A = ";: INPUT A
310  PRINT : PRINT "ENDING POINT B = ";: INPUT B
320  HOME
330  REM  DISPLAY FUNCTION ON SCREEN:
340  LIST 290
350  VTAB 2
360  PRINT "FOR FUNCTION"
370  PRINT : PRINT "STARTING POINT: A = ";A
380  PRINT : PRINT "ENDING POINT: B = ";B
390  PRINT : PRINT "N = NUMBER OF RECTANGLES"
400  PRINT : PRINT "TYPE 'S' TO STOP."
410  POKE 34,11
420  VTAB 12
```

```
430  REM  INITIALIZE N:
440 N = 1
450  PRINT "N = ";N;", ";
460  REM  INITIALIZE SUM:
470 SUM = 0
480  REM  COMPUTE STEP SIZE:
490 DX = (B - A) / N
500  REM   COMPUTE SUM OF FUNCTION VALUES:
510  FOR K = 1 TO N
520 SUM = SUM +   FN F(A + K * DX)
530  NEXT K
540  REM   MULTIPLY BY BASE LENGTH:
550 AN = SUM * DX
560  PRINT "APPROX. INTEGRAL = ";AN
570  GET ANS$
580  IF ANS$ = "S" THEN 620: REM  END
590  REM  GET NEW NUMBER N OF STEPS:
600 N = N * 2
610  GOTO 450
620  POKE 34,0
630  END
```

Lines 120 to 250 print a greeting and give you an opportunity to change the function F(X).

Lines 280 to 290 define the function.

Lines 300 to 310 obtain the end points A and B of the interval.

Lines 320 to 420 display the steup on the screen.

Lines 430 to 440 initialize N, the number of rectangles.

Lines 460–470 initialize SUM, which will hold the sum of the function values.

Lines 480 to 490 compute the step size DX for the current value of N.

Lines 510 to 530 contain a FOR-NEXT loop that computes the sum of the function values.

Lines 540 to 550 compute the rectangle area sum AN by multiplying the function value sum by the base length DX.

Line 560 prints the integral approximation AN.

Lines 570 to 580 pause to see if you want to do another approximation. If so, line 600 doubles the number N of rectangles and line 610 sends control back to line 450 for another round.

Exercise 30-1. Run INTEGRAL LIMIT for the given function F(X) = X and the end points A = 0, B = 3. Keep going for several rounds. What seems to be the limit of the approximations?

The function F(X) = X of Exercise 30-1 has a straight-line graph (see Figure 30-3), so we can use geometry to figure its integral and hence check your limit guess. Over the interval $0 \leq X \leq 3$, the graph of F(X) = X makes a right triangle with base length 3 and height 3. Hence the area is 3*3/2 = 4.5, which should agree with your limit guess in Exercise 30-1.

Exercise 30-2. Use INTEGRAL LIMIT to approximate the integral of F(X) = X over each of the following intervals. In each case, use area formulas of triangles or trapezoids to check that the rectangle approximations approach the correct area.

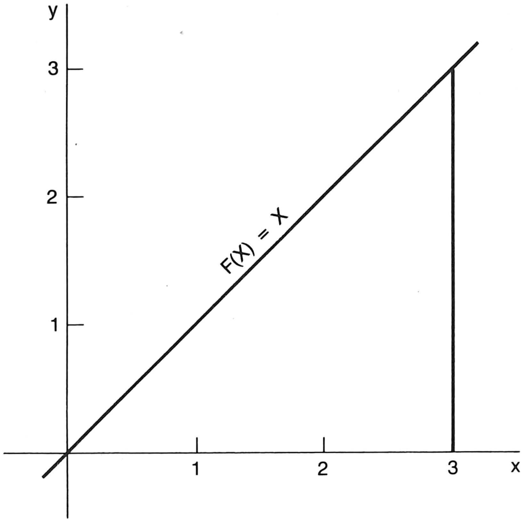


Figure 30-3 F(X) = X

1. A = 0, B = 4
2. A = 1, B = 2
3. A = 1, B = 3

Exercise 30-3. Repeat Exercise 30-2 for F(X) = 1 + 2X.

In Exercises 30-2 and 30-3, you worked with straight-line functions whose integrals you could compute geometrically. You saw that the rectangle rule did approach the true area value. Now let's look at a function whose integral you cannot compute geometrically.

Exercise 30-4. For the function $F(X) = X^2$, approximate the integral over each of the following intervals. Do enough steps so that you can be reasonably sure of the first decimal place in the limit. Can you guess the exact value of the limit? For future reference, write down the approximations you get for N = 512.

1. A = 0, B = 1
2. A = 0, B = 2
3. A = 1, B = 3

You see from Exercise 30-4 that the rectangle rule can take a long time to converge to a limit. We can speed things up by using a better approximation method. Rather than using rectangles to approximate the area under the curve, we'll use trapezoids. That is, we'll be using the *trapezoidal rule*. It is clear from Figure 30-4 that this should give a better approximation to the area.

The area of a trapezoid is the average (arithmetic mean) of the lengths of the parallel

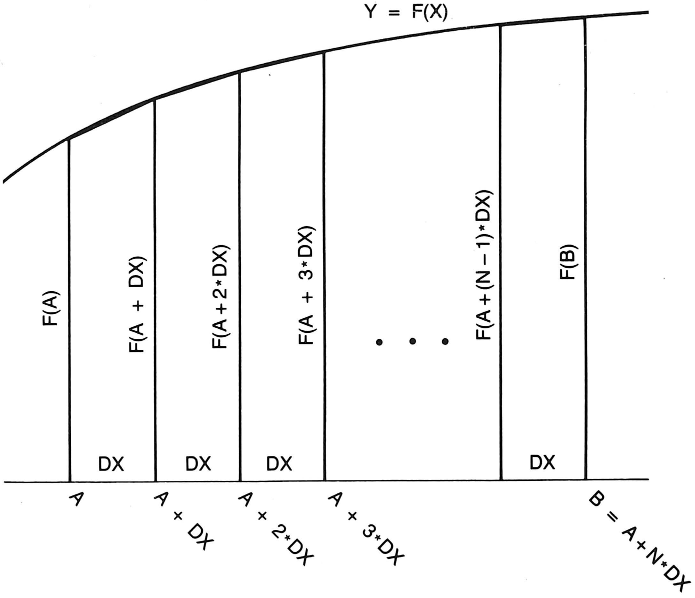


Figure 30-4 Trapezoids approximating area

sides times the distance between them. Here the distance between the parallel sides is always DX and the lengths of the parallel sides are given by the values of the function at the end points of the intervals. Following are the areas of the various trapezoids:

First: ½[F(A) + F(A + DX)]DX
Second: ½[F(A + DX) + F(A + 2*DX)]DX
Third: ½[F(A + 2*DX) + F(A + 3*DX)]DX
.
.
.
Last: ½{F[A + (N − 1)*DX] + F(B)}DX

Each of these areas has both ½ and DX as factors. Instead of adding these products as they stand, we can factor out the ½ and the DX from each term, add the results, and then multiply the sum by ½ and DX at the end. Here are the terms without the ½ and DX:

First: F(A) : F(A + DX)
Second: F(A + DX) + F(A + 2*DX)
Third: F(A + 2*DX) + F(A + 3*DX)

Last: F[A + (N − 1)*DX] + F(B)

When we add these up, we get

F(A) + 2*F(A + DX) + 2*F(A + 2*DX) + . . . + 2*F[A + (N − 1)*DX] + F(B)

because each term except F(A) and F(B) appears twice in the sum. We must still multiply this by ½ and DX to get the area of the trapezoids. When we multiply by ½, we get

$$\frac{F(A)}{2} + F(A + DX) + F(A + 2*DX) + \ldots + F[A + (N - 1)*DX] + \frac{F(B)}{2}$$

which is the same as

$$\frac{F(A) + F(B)}{2} + F(A + DX) + F(A + 2*DX) + \ldots + F[A + (N - 1)*DX]$$

So, for this part of the computation, we can initialize SUM to be

$$SUM = \frac{F(A) + F(B)}{2}$$

and then add in F(A + K*DX) for K = 1 to N − 1. The area of the trapezoids is then given by SUM*DX.

Exercise 30-5. Modify INTEGRAL LIMIT so that it uses the trapezoidal rule.

1. Change line 470 so that SUM is initialized as [F(A) + F(B)]/2.
2. Change line 510 so that K stops with N − 1.
3. Change "rectangle" to "trapezoid" in the print statements.

Exercise 30-6. Run this trapezoidal-rule version for F(X) = X, A = 0, B = 3, where you know the true value for the area is 4.5. How's that for fast convergence?

Exercise 30-7. Now let's do Exercise 30-4 again. Run your trapezoidal-rule version for $F(X) = X^2$ for each of the following intervals. How many steps does it take for you to be sure of the first decimal place in the limit? Compare your value for N = 512 here with the value you obtained in Exercise 30-4. Can you guess the limiting values here with more confidence than with the rectangle rule?

1. A = 0, B = 1
2. A = 0, B = 2
3. A = 1, B = 3

Exercises 30-6 and 30-7 show that the trapezoidal rule reveals the integral limit much more quickly than the rectangle rule. Hence, we will use the trapezoidal rule from now on.

Having investigated the integral as a limit, we now study some of the properties of the integral. The program INTEGRAL FUNCTION (see Listing 30-2) uses the trapezoidal rule with a fairly large number of trapezoids to approximate the integrals. It allows you to compute the integral of a given function for various upper end points B.

LISTING 30-2

```
100  REM       INTEGRAL FUNCTION
110  HOME
120  PRINT "    THIS PROGRAM USES THE TRAPEZOIDAL"
130  PRINT : PRINT "RULE TO APPROXIMATE THE INTEGRAL"
140  PRINT : PRINT "OF A FUNCTION Y = F(X) FROM"
150  PRINT : PRINT "X = A TO X = B."
160  PRINT : PRINT
170  PRINT : PRINT "THE FUNCTION IS SPECIFIED IN LINE 290:"
180  LIST 290
190  PRINT : PRINT : PRINT "DO YOU WISH TO CHANGE THE FUNCTION
?"
200  PRINT : PRINT "(Y OR N)";: INPUT ANS$
210  IF ANS$ <  > "Y" THEN 270
220  HOME
230  PRINT : PRINT "ENDING SO YOU CAN CHANGE THE FUNCTION:"
240  LIST 290
250  PRINT : PRINT "TO RESTART, TYPE 'RUN 270'"
260  END
270  HOME
280  REM  DEFINE FUNCTION:
290  DEF  FN F(X) = X * X
300  PRINT : PRINT "STARTING POINT A = ";: INPUT A
310  HOME
320  REM  DISPLAY FUNCTION ON SCREEN:
330  LIST 290
340  VTAB 2
350  PRINT "FOR FUNCTION"
360  PRINT : PRINT "STARTING POINT: A = ";A
370  PRINT : PRINT "TYPE 'S' TO STOP."
380  POKE 34,7
390  VTAB 8
400  REM   SET NUMBER OF TRAPEZOIDS:
410 N = 100
420  INPUT "B = ";B$
430  REM  EVALUATE B STRING:
440 B =  VAL (B$)
450  REM  SEE IF B$ WAS A NUMBER:
460  IF B <  > 0 THEN 490: REM   B$ WAS A NUMBER
470  REM   HERE, B=0, SO SEE IF B$ WAS '0'
480  IF B$ <  > "0" THEN 620: REM   END
490  REM  COMPUTE INTEGRAL
500  REM  INITIALIZE SUM:
510 SUM = ( FN F(A) +  FN F(B)) / 2
520  REM  COMPUTE STEP SIZE:
530 DX = (B - A) / N
540  REM   COMPUTE SUM OF FUNCTION VALUES:
550  FOR K = 1 TO N - 1
560 SUM = SUM +  FN F(A + K * DX)
570  NEXT K
580  REM   MULTIPLY BY BASE LENGTH:
590 AN = SUM * DX
600  PRINT "       APPROX. INTEGRAL = ";AN
610  GOTO 420
620  POKE 34,0: REM  END SCROLLING PROTECTION
630  END
```

Lines 100 to 290 print a greeting and give you the opportunity to change the function, which is defined in line 290.

Line 300 obtains the lower end point A.

Lines 310 to 390 print the setup on the screen.

Lines 400 to 410 set the number N of trapezoids to 100.

Line 420 asks for the upper end point B. The response is entered as a string so that the program can check if you typed ''S''. If you did type ''S'', or any nonnumber symbol, the program ends. If you typed a number for B, lines 490 to 610 compute and print the integral and return to line 420 for another B.

Use INTEGRAL FUNCTION in the following exercises.

Exercise 30-8. Approximate the integral of the function $F(X) = X^2$ over the following intervals. Compare the results with the results obtained in Exercise 30-4 to see that you're getting the correct answers.

1. $A = 0,\ B = 1$
2. $A = 0,\ B = 2$
3. $A = 1,\ B = 3$

Now let's investigate some of the properties of the integral. For example, we assumed at the start of this chapter that our function F(X) was positive over the interval from A to B. However, the limit idea we've been using to compute the integral makes sense even for nonpositive functions. So, let's see what the integral gives for a negative function.

Exercise 30-9. Compute the integral of $F(X) = -X^2$ over the same intervals as in Exercise 30-8. Compare your answers here with those for $F(X) = X^2$ in Exercise 30-8.

Exercise 30-9 shows that the integral of a negative function does not give the area between the curve and the X axis, because the *area* of a region must be a positive number. Rather, the integral of such a function gives the opposite (additive inverse) of the area.

Moreover, Exercise 30-9 shows that changing the sign of the function changes the sign of the integral:

$$\int_B^A -F(X)DX = -\int_B^A F(X)DX$$

Now that we have connected integrals and area for positive and for negative functions, what about a function that is positive over one part of the interval and negative over another part? For example, the function $F(X) = 1 - X^2$ is positive for $0 < X < 1$ and negative for $1 < X < 2$. (See Figure 30-5.) The next exercise investigates this function.

Exercise 30-10. Compute the following integrals

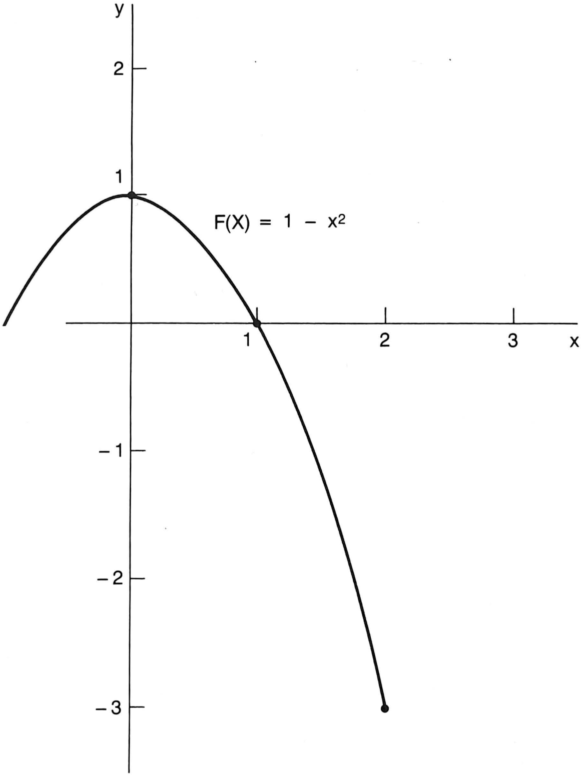


Figure 30-5 F(X) = 1-X^2

1. $\int_0^1 1 - X^2\, DX$
2. $\int_1^2 1 - X^2\, DX$
3. $\int_0^2 1 - X^2\, DX$

You see from this exercise that, for $A < B < C$, the integral satisfies the property

$$\int_A^C F(X)DX = \int_A^B F(X)DX + \int_B^C F(X)DX$$

In terms of areas, Exercise 30-10 shows that the integral over the entire interval gives the area of the region above the X axis minus the area of the region below the axis.

Exercise 30-11. Compute the integral

$$\int_2^2 X^2\, DX$$

This exercise illustrates the property

$$\int_A^A F(X)DX = 0$$

Exercise 30-12.

1. Compute the integral of $F(X) = X + X^2$ over the interval from $A = 0$ to $B = 2$. Write down your answer.
2. Compute the integral of $F(X) = X + X^2$ again, but this time enter the end points in reverse order. That is, enter $A = 2$ and $B = 0$.
3. Compare your results in parts a) and b). What general property of integrals does your comparison suggest?

Exercise 30-13.

1. Compute the integral of sin(X) over the interval $-1 \leq X \leq 2$.
2. Compute the integral of cos(X) over the interval $-1 \leq X \leq 2$.
3. Compute the integral of sin(X) + cos(X) over the interval $-1 \leq X \leq 2$.
4. State a general property of integrals illustrated by parts 1 through 3.

It would be nice to have formulas for evaluating integrals. For example, we could check up on the computer's approximations. To see how to get such formulas, let's hold the lower end point A fixed and compute the integral for various upper end points B.

Exercise 30-14. For the function $F(X) = X$ and lower end point $A = 0$, compute the integral for each of the following end points B. Record your answers.

1. $B = 1$
2. $B = 2$
3. $B = 3$
4. $B = 4$
5. $B = 5$
6. $B = 6$

Try to find a formula for the integral values in terms of B. It may help you to see the formula if you double each of the integral values.

Exercise 30-15. For the function $F(X) = X^2$, use starting point $A = 0$ and compute the integral for each of the following ending points B. Record your answers.

1. $B = 1$
2. $B = 2$

3. B = 3
4. B = 4
5. B = 5
6. B = 6

Try to find a formula for these integral values in terms of B. It may help here to multiply the integrals by 3.

So far, we've been using A = 0 as base point. What happens if we use a different base point? Try the next two exercises to see what occurs.

Exercise 30-16. Again using the function F(X) = X, use lower end point A = 2, and compute the integral for each of the following upper end points B. Record your answers.

1. B = 1
2. B = 2
3. B = 3
4. B = 4
5. B = 5
6. B = 6

Find a formula for the integrals in terms of B. It may help to refer to your integral values in Exercise 30-14.

Exercise 30-17. For the function F(X) = X^2 and base point A = 1, compute the integral for each of the following end points B. Record your answers.

1. B = 1
2. B = 2
3. B = 3
4. B = 4
5. B = 5
6. B = 6

Find a formula for these integrals in terms of B. It may help to compare these integrals with those in Exercise 30-15.

To get the integral values in Exercise 30-16 from those in Exercise 30-14, you just have to subtract 2. For example in Exercise 30-14, B = 4 gives an integral value of 8. In Exercise 30-16, B = 4 gives 6. Since the function that gave the integrals in Exercise 30-14 was $B^2/2$, in Exercise 30-16 it must be

$$\frac{B^2}{2} - 2 \qquad (30\text{-}1)$$

How does this value depend on the base point A? The base point in Exercise 30-16 is A = 2, and $2^2/2 = 2$, so we can write the function (30-1) as

$$\frac{B^2}{2} - \frac{A^2}{2} \qquad (30\text{-}2)$$

Note that this formula also works for Exercise 30-14, where A = 0.

Now let's see if a similar thing happens in Exercise 30-17. To get the integrals in Exercise 30-17 from those of Exercise 30-15, you just have to subtract ⅓. For example, B = 3 in Exercise 30-15 gives 9.00045 and B = 3 in Exercise 30-17 gives 8.66679999. Since the integral function in Exercise 30-15 is $B^3/3$, the function in Exercise 30-17 must be

$$\frac{B^3}{3} - \frac{1}{3} \qquad (30\text{-}3)$$

The base point in Exercise 30-17 is A = 1, and $1^3/3$ = ⅓, so we can write this function as

$$\frac{B^3}{3} - \frac{A^3}{3} \qquad (30\text{-}4)$$

Thus, we have found two integral formulas:

$$\int_A^B X \, DX = \frac{B^2}{2} - \frac{A^2}{2} \qquad (30\text{-}5)$$

$$\int_A^B X^2 \, DX = \frac{B^3}{3} - \frac{A^3}{3} \qquad (30\text{-}6)$$

The next two exercises give you a chance to test formulas (30-5) and (30-6).

Exercise 30-18. Insert a line in INTEGRAL FUNCTION so that it prints the value of

$$\frac{B^2}{2} - \frac{A^2}{2}$$

after each integral computation. Run this version for F(X) = X and various choices for A and B. Do the formula values agree with the computer's approximate integral values?

Exercise 30-19. Modify your program so that it prints the value of

$$\frac{B^3}{3} - \frac{A^3}{3}$$

after each integral computation. Run this version for $F(X) = X^2$ and various choices of A and B. Does the formula work?

There is a notation that simplifies the writing of integral formulas such as (30-5) and (30-6). In (30-5), for example, the formula involves the function $X^2/2$. To get the value of the integral, you evaluate this function at B and at A and substract. For a general function G(X), we use the following shorthand for this evaluation and substraction:

$$G(X) \,|_A^B = G(B) - G(A)$$

With this notation, the integral formulas (30-5) and (30-6) become

$$\int_A^B X \, DX = \frac{X^2}{2} \,\Big|_A^B \tag{30-7}$$

$$\int_A^B X^2 \, DX = \frac{X^3}{3} \,\Big|_A^B \tag{30-8}$$

Note the calculus connection between the functions in these formulas: For $F(X) = X$, the integral function is $X^2/2$. But the derivative of $X^2/2$ is X. So, the derivative of the integral function is the original function! The same thing happens with the other example: The original function is X^2, and the integral function is $X^3/3$. Again, the derivative of the integral function $X^3/3$ is the original function X^2.

These examples suggest that to compute the integral of a function F(X) from formulas, you must find an *antiderivative* G(X) for F(X), that is, a function G(X) whose derivative is F(X). This procedure is made explicit in the following theorem.

Fundamental Theorem of Calculus: If G(X) is an antiderivative of a continuous function F(X),

$$\int_A^B F(X) DX = G(X) \,\|_A^B$$

This theorem is fundamental to calculus because it connects the two limit ideas—the derivative and the integral. It also gives you an alternative to using the computer's approximations for integrals. You can find an antiderivative and use it to get the integral. In the next exercise, you check the theorem for various functions.

Exercise 30-20. Listed below are pairs of functions F(X) and antiderivatives G(X). For each pair, insert a line in INTEGRAL FUNCTION to compute G(B) − G(A). Then compare the integral value to the value of G(B) − G(A) for various choices of A and B to see if the fundamental theorem seems to hold.

1. $$F(X) = X, \; G(X) = \frac{X^2}{2}$$

2. $$F(X) = X, \; G(X) = \frac{X^2}{2} + 5$$

3. $$F(X) = X^3,\ G(X) = \frac{X^4}{4}$$

4. $$F(X) = \sin(X),\ G(X) = -\cos(X)$$

5. $$F(X) = \cos(X),\ G(X) = \sin(X)$$

6. $$F(X) = \frac{1}{(X^2 + 1)},\ G(X) = \arctan(X)$$

7. $$F(X) = \frac{1}{X},\ G(X) = \log(X)$$

Computing integrals by using antiderivatives has the advantage of yielding the exact value of the integral rather than an approximation such as the trapezoidal rule gives. On the other hand, it is often difficult or impossible to find a formula for the antiderivative of a function. For example, the function

$$F(X) = \exp\left(\frac{-X^2}{2}\right)$$

gives the bell-shaped normal curve used in statistics. To use the normal curve in statistics, you must be able to compute the integral of this function between any two points. However, no formula exists for an antiderivative for this function. So you have to use an approximation method such as the trapezoidal rule.

Exercise 30-21. Use INTEGRAL FUNCTION to compute the integral of the function $\exp(-X^2/2)$ over various intervals.

31

ROOT SEARCHES

In Chapters 11 and 12, we studied ways of approximating square roots by fractions. Here we study approximation methods that are closer to what the computer probably does in its SQR function. Why does the machine have to use an approximation technique to get values for square roots? Well, all of the machine's internal computations are based on addition: It multiplies by adding repeatedly. It subtracts by adding complements and divides by subtracting repeatedly. So how does it compute square roots? That is, how can it reduce the approximating of square roots to the operations of addition, subtraction, multiplication, and division?

In this chapter we will study two methods of approximating roots of the kind the computer might use. These methods apply to finding general roots of functions, not just square roots. By a *root* of a function F(X) we mean a number X such that F(X) = 0. For example, the function $F(X) = X^2 - 3$ has two roots, $X = \sqrt{3}$ and $X = -\sqrt{3}$. So, if we can find a way to approximate the roots of a general function F(X), this example shows that we can certainly approximate square roots.

We first study the *bisection method* of searching for a root of a function. This method is based on the following fact: If a function F(X) is continuous on an X interval from XL to XH, and if YL = F(XL) and YH = F(XH) have different signs, then F(X) must be zero for some X between XL and XH. (See Figure 31-1.) That is, if either YL or YH is negative and the other is positive, there must be a root X somewhere between XL and XH.

To close in on the root X, bisect the interval. The midpoint of the interval is XM = (XL + XH)/2. Now compute YM = F(XM). If YM = 0, we have found a root at XM. If not, either YM and YL have different signs, or YM and YH do.

If YM and YL have different signs (as in Figure 31-1), make XM the new right-

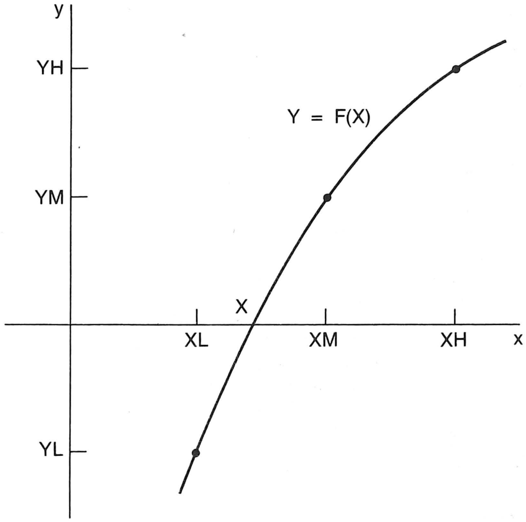


Figure 31-1 Bisection method

hand end point (that is, let XH = XM) and bisect again. If YM and YH have different signs, make XM the new left-hand end point (by letting XL = XM) and bisect again.

At each stage, we know the root X lies between XL and XH. Furthermore, XL and XH are getting closer together, so they are getting closer and closer to the root. In fact, since the root is between XL and XH, if we take either XL or XH as an estimate for the root, the error is less than the length XH − XL of the interval.

The program BISECTION (see Listing 31-1) uses the bisection method to approximate square roots.

LISTING 31-1

```
100  REM      BISECTION
110  HOME
120  PRINT "     THIS PROGRAM APPROXIMATES SQUARE"
130  PRINT : PRINT "ROOTS BY THE BISECTION METHOD."
140  PRINT : PRINT
150  PRINT : PRINT "     FOR WHAT NUMBER N DO YOU WANT TO"
160  PRINT : PRINT "APPROXIMATE THE SQUARE ROOT";
170  INPUT N
180  REM  SET INITIAL INTERVAL:
190  IF N >  = 1 THEN XL = 1:XH = N
200  IF N < 1 THEN XL = N:XH = 1
210  REM  DEFINE FUNCTION:
220  DEF  FN F(X) = X * X - N
230  REM  INTIALIZE COUNTER FOR NUMBER OF BISECTIONS:
240 COUNTER = 0
250  HOME
260  PRINT "HIT 'S' TO STOP, 'RETURN' TO CONTINUE"
270  PRINT : PRINT "APPROXIMATING SQUARE ROOT OF ";N
280  PRINT
290  PRINT "  XL                XH"
300  VTAB 7
310  POKE 34,6: REM  PROTECT INSTRUCTIONS FROM SCROLLING
320  REM   PRINT INITIAL INTERVAL ENDS:
```

```
330  PRINT XL,XH
340  REM  PAUSE:
350  GET ANS$
360  REM   COMPUTE Y VALUES AT XL,XH:
370 YL =  FN F(XL)
380 YH =  FN F(XH)
390  REM  COMPUTE INTERVAL MID-POINT:
400 XM = (XL + XH) / 2
410  REM  UPDATE COUNTER:
420 COUNTER = COUNTER + 1
430  REM  COMPUTE Y AT MID-POINT:
440 YM =  FN F(XM)
450  REM  SEE IF MID-POINT IS A ROOT:
460  IF YM = 0 THEN 610: REM  END
470  REM     SEE IF YL AND YM HAVE DIFFERENT SIGNS
480  REM     IF SO, THE ROOT IS BETWEEN XL,XM,
490  REM       SO SET THE NEW XH = XM:
500  IF YL * YM < 0 THEN XH = XM:YH = YM: GOTO 540
510  REM      HERE, THE ROOT IS BETWEEN XM,XH.
520  REM      SO SET THE NEW XL = XM:
530 XL = XM:YL = YM
540  REM  PRINT INTERVAL END POINTS:
550  PRINT XL,XH
560  REM   PAUSE:
570  GET ANS$
580  REM    STOP IF OPERATOR HIT "S":
590  IF ANS$ = "S" THEN 630: REM   END
600  GOTO 390: REM      BISECT AGAIN
610  REM  HERE, HAVE HIT ROOT
620  PRINT : PRINT "ROOT AT X = ";XM
630  PRINT : PRINT "NUMBER OF BISECTIONS = ";COUNTER
640  PRINT : PRINT "XL*XL = ";XL * XL
650  PRINT : PRINT "XH*XH = ";XH * XH
660  PRINT : PRINT "INTERVAL LENGTH XH-XL = ";XH - XL
670  PRINT : PRINT "DO ANOTHER ONE (Y OR N)";
680  INPUT ANS$
690  POKE 34,0: REM  END SCROLLING PROTECTION
700  IF ANS$ = "Y" THEN  HOME : GOTO 150
710  END
```

Lines 100 to 170 print a greeting and obtain the number N whose square root is to be approximated.

To start the bisection method, we need an interval XL to XH such that XL $< \sqrt{N}$ and XH $> \sqrt{N}$. If $1 < N$, then $1 < \sqrt{N}$, so we can take XL = 1. For a right-hand end point XH, we can take N, because $\sqrt{N} < N$, if $N > 1$. On the other hand, if $N < 1$, then $\sqrt{N} < 1$, so we can take XH = 1. For XL, we can take N, because $N < \sqrt{N}$, for $N < 1$. These initial values for XL and XH are set in lines 180 to 200.

To approximate $\sqrt{N}$, we use the function $F(X) = X^2 - N$. Note that the DEF statement in line 220 uses multiplication to compute X^2, so we are using only arithmetic to compute the function.

Lines 230 to 240 initialize a counter for the number of bisections.

Lines 250 to 310 print instructions on the screen and protect them from scrolling.

Lines 320 to 330 print the initial low and high estimates XL and XH.

Lines 340 to 350 pause before lines 360 to 380 compute the function values at the end points.

Lines 390 to 400 compute the midpoint XM.

Lines 410 to 420 update the counter to account for this new bisection.

Lines 430 to 440 compute the function value YM at the midpoint XM.

If XM happens to be a root, line 460 sends control to the ending ceremony.

Otherwise, line 500 tests whether YL and YM have different signs. If so, XM is made the new right-hand end point. If not, XM is made the new left-hand end point in line 530.

Lines 540 and 550 print the new interval end points.

Lines 560 to 600 pause and give you the chance to end the bisection process. If you choose to end, lines 630 to 660 print the number of bisections, the squares of the low and high estimates, and the maximum error XH-XL.

Finally, lines 670 to 700 give you a chance to do another root.

Exercise 31-1. Use BISECTION to approximate the square roots of each of the following numbers N. Do several steps. Check the initial end points: Are they 1 and N? Look at how the end points change. Is each new end point halfway between the old ones? When you stop bisecting, are the estimates close to the square root? That is, are their squares close to N?

1. N = 2
2. N = 47
3. N = 0.5
4. N = 0.123
5. N = 49
6. N = 465

Exercise 31-2. Run BISECTION to approximate the square root of each of the following numbers N. Do five steps for each number. After the five bisections, how close are the estimates XL and XH to the true value of the root? How close are they after seven steps?

1. N = 3
2. N = 9
3. N = 0.454
4. N = 778

Exercises 31-1 and 31-2 show that the bisection method does indeed give good estimates for square roots and can even tell you how close the estimates are to the true value of the root. Now let's push this method to the accuracy limit of the computer.

Exercise 31-3. Approximate the square root of each of the following numbers. Do enough steps so that XL and XH agree to seven decimal places. Note how many steps it takes and what the maximum error is when you stop.

1. N = 5
2. N = 0.333
3. N = 47.34

An automatic approximation program, such as the computer uses for its SQR function, must do several approximation steps and stop when a fairly accurate estimate is reached. To modify BISECTION so that it does the approximation automatically, we just need to eliminate the pause between bisections and have the bisecting stop when the interval length XH − XL is small enough to give the accuracy we want. The next exercise asks you to make these changes.

Exercise 31-4.

1. Insert a line to input the error bound:

```
175 PRINT: PRINT "ERROR BOUND = "; : INPUT BOUND
```

2. Remove the pausing between bisections by deleting lines

```
560 REM PAUSE:
570 GET ANS$
580 REM STOP IF OPERATOR HIT "S":
590 IF ANS$ = "S" THEN 630: REM END
```

3. Insert a line to make the bisecting stop when the maximum possible error is less than BOUND:

```
560 IF XH-XL < BOUND THEN 630: REM END
```

4. Insert a line in the ending to report the error bound:

```
665 PRINT: PRINT "ERROR BOUND = "; BOUND
```

Exercise 31-5. Run your modified version of BISECTION to approximate the square roots of the following numbers N. Use an error bound of 10^{-4} to check that the bisecting is stopping when it should. In BASIC you type 1E-4 for 10^{-4}.

1. N = 3
2. N = 7
3. N = 0.345
4. N = 0.123
5. N = 77
6. N = 576

7. N = 8,796
8. N = 45,786

What is the best we can expect from the bisection method? If the estimates XL and XH agree in all decimal places except the last and differ by only 1 in that place, they are as close together as different computer numbers can be. For example, if XL = 4.78564352 and XH = 4.78564353, the computer cannot fit any other number between these two. (Here we are talking about what the computer shows on the screen, which, as you know, may be different from what it holds inside.) To get the most out of the bisection method, then, we just have to choose the error bound small enough so that XL and XH differ by just 1 in their last decimal place when the bisection stops.

Exercise 31-6. Do Exercise 31-5 again, this time start with error bound 1E-8. For each N, write down the number of bisections and whether the final estimates are as close together as possible. If they are not, find an error bound that makes them so.

Exercise 31-6 shows that choosing the correct error bound is not trivial. If the number is close to 0, you need a smaller error bound. If the number is large, you need a larger bound. Moreover, if the error bound is too small for N, the program gets into an infinite loop. So, if the computer uses bisection in its SQR function, it must have some way of choosing the correct error bound based on the size of N.

So far, we've been avoiding comparing our estimates to SQR(N) to convince you that we can indeed approximate square roots on our own. Now let's compare our results with SQR(N).

Exercise 31-7. Modify your current BISECTION program so that it prints SQR(N) and its square. Insert lines

```
666 PRINT:PRINT "SQR(";N;") = ";SQR(N)
667 PRINT:PRINT "AND ITS SQUARE IS ";SQR(N)*SQR(N)
```

Run this latest version of BISECTION for the numbers N of Exercise 31-5 and various error bounds. Check that XL and XH are close to SQR(N) if the error bound is small. Now let's see if bisection gives as good answers as the computer's SQR(N).

Exercise 31-8. Approximate the square roots of the numbers N of Exercise 31-5. Use the error bounds you found in Exercise 31-6 which gave XL and XH as close together as possible. Does XL or XH equal SQR(N)? Do they all square to yield N?

Exercise 31-9. Approximate the square roots of the following numbers N with the given error bounds.

1. N = 829.23, error bound = 2E-8
2. N = 5467, error bound = 1E-7

Surprised? In these examples, the bisection method is *better* than SQR(N)! In each of these cases, one of XL or XH squares to give exactly N, whereas SQR(N) does not.

Exercise 31-10. Try to find more numbers N for which the bisection method is better than SQR(N).

We now leave bisection and turn to another root approximation method, *Newton's method*. Let F(X) be the function and XO a guess at a root X (see Figure 31-2). At the point (XO, YO) on the curve, draw the tangent line. The point XN where the tangent line crosses the X axis should be closer to the root X than XO. Now repeat this: Make XN the guess (that is, let XO = XN) and do the tangent line maneuver again.

To carry out this method, we have to compute XN from XO. The point XN is where the tangent line from (XO, YO) hits the X axis, so we have two points on the line—(XO, YO) and (XN, 0). The slope M of the line is

$$\begin{aligned} M &= \frac{YO - 0}{XO - XN} \\ &= \frac{YO}{XO - XN} \end{aligned}$$

Multiplying both sides by XO − XN gives

$$\begin{aligned} YO &= M*(XO - XN) \\ &= M*XO - M*XN \end{aligned}$$

We want to compute XN, so solve this for XN. Adding M*XN to both sides and subtracting YO gives

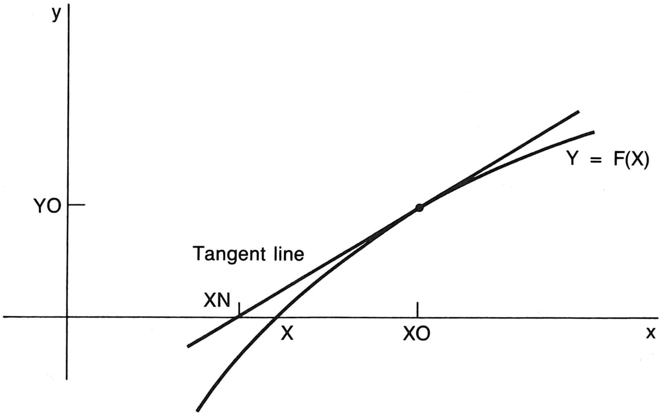


Figure 31-2 Newton's method

$$M*XN = M*XO - YO$$

Then dividing by M gives

$$XN = XO - \frac{YO}{M}$$

This shows how to compute XN from XO and YO, if we know the slope M of the tangent line and $M \neq 0$. What, then, is the slope of the tangent line? The derivative F′(XO), of course. Thus, if

$$YP = F'(XO)$$

and

$$XN = XO - \frac{YO}{YP} \tag{31-1}$$

XN is the next guess in Newton's method.

The program NEWTON'S (see Listing 31-2) applies Newton's method to the computing of square roots.

LISTING 31-2

```
100  REM      NEWTON'S
110  HOME
120  PRINT "    THIS PROGRAM APPROXIMATES SQUARE"
130  PRINT : PRINT "ROOTS BY NEWTON'S METHOD."
140  PRINT : PRINT
150  PRINT : PRINT "      FOR WHAT NUMBER DO YOU WANT TO"
160  PRINT : PRINT "APPROXIMATE THE SQUARE ROOT";
170  INPUT N
180  HOME
190  REM  SET INITIAL GUESS:
200 XO = N
210  REM  INITIALIZE COUNTER FOR NUMBER OF STEPS:
220 COUNTER = 0
230  REM     DEFINE FUNCTION:
240  DEF  FN F(X) = X * X - N
250  REM  DEFINE DERIVATIVE FUNCTION:
260  DEF  FN DF(X) = 2 * X
270  HOME
280  PRINT "HIT 'S' TO STOP, 'RETURN' TO CONTINUE."
290  PRINT : PRINT "   X","     X*X"
300  PRINT
310  PRINT XO,XO * XO
320  REM      COMPUTE Y COORDINATE:
330 YO =  FN F(XO)
340  REM  IF YO = 0, HAVE FOUND ROOT
350  IF YO = 0 THEN 490: REM  END
360  REM  PAUSE:
370  GET ANS$
380  REM  IF OPERATOR HIT "S" THEN END:
390  IF ANS$ = "S" THEN 500
400  REM  COMPUTE SLOPE OF TANGENT LINE:
410 YP =  FN DF(XO)
```

```
420  REM      X PT WHERE TANGENT   HITS X AXIS:
430 XN = XO - YO / YP
440  REM  UPDATE COUNTER FOR NUMBER OF STEPS:
450 COUNTER = COUNTER + 1
460  REM      UPDATE X:
470 XO = XN
480  GOTO 310: REM    DO IT AGAIN
490  PRINT : PRINT "ROOT AT XO = ";XO
500  PRINT : PRINT "NUMBER OF STEPS = ";COUNTER
510  PRINT : PRINT "COMPUTER'S SQR(";N;") = "; SQR (N)
520  PRINT : PRINT "    ITS SQUARE = "; SQR (N) *  SQR (N)
530  PRINT : PRINT "DO ANOTHER ONE";
540  INPUT ANS$
550  IF ANS$ = "Y" THEN  HOME : GOTO 150
560  END
```

Lines 100 to 170 print a greeting and input the number N whose square root is to be approximated.

Newton's method needs an initial guess XO to get started, and line 200 sets XO rather arbitrarily to N.

Lines 210 to 220 initialize a counter for the number of steps.

Lines 230 to 240 define the function F(X) = $X^2 - N$, which has $\sqrt{N}$ as a root.

Lines 250 to 260 tell the computer the formula for the derivative function, DF(X) = 2X.

Lines 270 to 300 print some instructions and a heading.

Line 310 prints XO and XO^2 so you can see the root estimate and its square.

Lines 320 and 330 compute the Y coordinate YO.

Line 350 sends control to the ending part if XO happens to be a root.

If XO is not a root, lines 360 to 390 pause and give you the chance to end the approximation. If you do not choose to end, lines 400 to 410 compute the slope YP of the tangent line.

Line 430 computes the new guess XN according to the formula (31-1) we derived above.

Lines 440 to 450 update the counter and line 470 makes XN the next guess. Then line 480 loops back for another round.

In the ending ceremony, lines 500 to 560, the program prints the number of steps, the computer's value for SQR(N), and its square. Then it gives you the chance to approximate the root of another number.

Exercise 31-11. Run NEWTON'S to approximate the square roots of various numbers N. For each N, do enough steps before stopping to have a good estimate for the root. Then compare your estimate to SQR(N). Does this method converge to the root as fast as the bisection method?

Newton's method has a serious disadvantage compared with the bisection method. With Newton, you have no way of knowing how close a given estimate is to the root, nor whether the estimate is too large or too small. On the other hand, it converges so fast that the computer may be using it for SQR anyway. To use Newton's method in an automatic approximation program such as SQR, we must be able to shut it off when we think we have a good estimate. There are two ways to do this, when (XN)^2 is close to N, or when the new guess XN is close to the old guess XO. Here we'll try the second method. You can experiment with the first on your own.

Exercise 31-12. Modify NEWTON'S so that it stops when |XN − XO| is smaller than 10^{-7}:

1. Delete the pausing lines

```
360 REM PAUSE:
370 GET ANS$
380 REM IF OPERATOR HIT "S" THEN END:
390 IF ANS$= "S" THEN 500
```

2. Insert a new ending test:

```
455 IF ABS(XN-XO) < 1E-7 THEN 500:REM END
```

3. Run this version to approximate the square roots of various numbers.

Exercise 31-13. Approximate the square roots of the numbers in Exercise 31-5 with this version of NEWTON'S. Compare the number of steps required by Newton's method with that required by the bisection method in Exercise 31-6.

You can see from working with Newton's method that it is much faster than the bisection method. Is this method the one the computer uses for SQR?

So far you have been approximating only square roots. You can just as easily approximate cube roots. To approximate the cube root of a number N, you can use the function $F(X) = X^3 - N$. The computer has a built-in exponentiation operation, "$\wedge$," so the computer would do cube roots by computing X $\wedge$ (⅓).

Exercise 31-14. Modify your latest bisection method program (from Exercise 31-7) so that it approximates cube roots:

1. Change the function to F(X) = X*X*X − N.
2. At the end of the bisections, print the cubes of XL and XH as products (don't use "$\wedge$") and print N $\wedge$ (⅓) rather than SQR(N).
3. Run this cube-root approximation program for various numbers N.
4. Can you find numbers N for which the bisection method gives a better cube root than N $\wedge$ (⅓)?

Exercise 31-15. Modify your latest Newton's method program (from Exercise 31-12) to do cube roots:

1. Change the function to F(X) = X*X*X − N.
2. Change the derivative function to DF = 3*X*X.

3. At each step, print XO and XO*XO*XO.
4. At the end, print N /\ (⅓) rather than SQR(N).
5. Run this Newton's cube-root program for various numbers N. Does it ever give better results than N /\ (⅓)?

Now that you have conquered cube roots, how about general Kth roots? For a fixed K, you can certainly approximate Kth roots by using the function F(X) = X*X* . . . . *X, where X is multiplied K times. But the computer's ''/\'' operation allows you to choose K, and then it approximates the root for whatever K you chose. If you want to write a program that allows the operator to choose K, compute F(X) with a FOR-NEXT loop to multiply X the correct number of times.

Exercise 31-16. Modify the bisection or Newton's method programs to do general Kth roots. This requires major surgery on the programs because you cannot use the DEF statements for the functions. Instead, you must use a subroutine with a FOR-NEXT loop to compute the function values.

If you were able to complete Exercise 31-16, you have a program that nearly mimics the computer's ''/\'' operation. Since the computer works only with terminating decimals, you need to compute only $X^{A/B}$, where A and B are integers. If you require A and B to be positive, you can use your Kth root program to compute $X^{1/B}$ and then multiply that by itself A times.

Exercise 31-17. Extend your program of Exercise 31-16 to compute $X^{A/B}$ for positive A and B. Compare with the computer's X /\ (A/B).

Now let's move on up from Kth roots of numbers N to general roots of functions. For example, you can use the bisection or Newton's method to approximate the roots of a polynomial such as

$$F(X) = 14X^3 - 27X^2 - 131X + 60 \tag{31-2}$$

One added difficulty here is getting the methods started. For bisection, you must have an interval XL to XH which contains a root. For Newton's, you must have an initial guess at a root. When approximating roots of numbers, it was fairly easy to make these starting choices. For a general function, however, it may not be so easy. So, it helps to have a program that computes a table of function values. From such a table, you may be able to find the necessary starting information. The program BISECTION GENERAL (see Listing 31-3) contains a subroutine for making a table of function values and a subroutine for applying the bisection method.

LISTING 31-3

```
100  REM   BISECTION GENERAL
110  DEF  FN F(X) = 14 * X ^ 3 - 27 * X ^ 2 - 131 * X + 60
120  HOME
130  PRINT "      THIS PROGRAM CAN COMPUTE A TABLE"
140  PRINT : PRINT "OF FUNCTION VALUES AND APPLY THE"
150  PRINT : PRINT "BISECTION METHOD TO SEARCH FOR A"
160  PRINT : PRINT "ROOT OF THE FUNCTION."
```

```
 170  PRINT : PRINT "    THE FUNCTION IS DEFINED IN LINE 110:"
 180  LIST 110
 190  PRINT : PRINT "DO YOU WANT TO CHANGE THE FUNCTION?"
 200  PRINT : PRINT "(Y OR N)";
 210  INPUT ANS$
 220  IF ANS$ <  > "Y" THEN 2000: REM   MENU SUBROUTINE
 230  HOME
 240  PRINT : PRINT "ENDING SO YOU CAN CHANGE THE FUNCTION:
 250  LIST 110
 260  END
 500  REM   TABLE SUBROUTINE
 510  HOME
 520  PRINT "STARTING X VALUE = ";
 530  INPUT X0
 540  PRINT : PRINT "X STEP SIZE = ";
 550  INPUT DX
 560  HOME
 570  PRINT "   X            F(X)"
 580  FOR K = 0 TO 20
 590 X = X0 + K * DX
 600  PRINT X, FN F(X)
 610  NEXT
 620  PRINT : PRINT "PRESS ANY KEY TO CLEAR SCREEN.";
 630  GET ANS$
 640  RETURN
1000  REM  BISECTION SUBROUTINE
1010  HOME
1020  PRINT "ENTER STARTING INTERVAL XL <= X <= XH."
1030  PRINT : PRINT "XL = ";
1040  INPUT XL
1050  PRINT : PRINT "XH = ";
1060  INPUT XH
1070  PRINT : PRINT "   XL            XH"
1080  PRINT
1090 YL =  FN F(XL)
1100 YH =  FN F(XH)
1110 XM = (XL + XH) / 2
1120 YM =  FN F(XM)
1130  IF YM = 0 THEN 1210
1140  IF YL * YM < 0 THEN XH = XM:YH = YM: GOTO 1160
1150 XL = XM:YL = YM
1160  PRINT XL,XH
1170  REM  PAUSE:
1180  GET ANS$
1190  IF ANS$ = "S" THEN 1220
1200  GOTO 1110
1210  PRINT : PRINT "ROOT AT X = ";XM
1220  PRINT : PRINT "F(";XL;") = "; FN F(XL)
1230  PRINT : PRINT "F(";XH;") = "; FN F(XH)
1240  PRINT : PRINT "PRESS ANY KEY TO CLEAR SCREEN."
1250  GET ANS$
1260  RETURN
2000  REM  MENU SUBROUTINE
2010  HOME
2020  PRINT "THE FUNCTION IS "
2030  LIST 110
2040  VTAB 3
2050  PRINT "                "
2060  VTAB 10
2070  PRINT "TO CHANGE FUNCTION OR STOP, TYPE 'S'
2080  PRINT : PRINT "TO MAKE A TABLE OF FUNCTION VALUES,"
2090  PRINT : PRINT "      TYPE 'T'"
2100  PRINT : PRINT "TO USE BISECTION METHOD, TYPE 'B'"
2110  PRINT : PRINT "WHICH DO YOU WANT TO DO";
2120  INPUT ANS$
2130  IF ANS$ = "S" THEN  END
2140  IF ANS$ = "T" THEN  GOSUB 500: GOTO 2000
2150  IF ANS$ = "B" THEN  GOSUB 1000: GOTO 2000
2160  PRINT : PRINT "I DON'T KNOW HOW TO ";ANS$
2170  GOTO 2110
```

Line 110 defines the function F(X).

Lines 120 to 260 print a description of the program and give you a chance to change the function. If you don't want to change the function, control is sent to the menu subroutine, lines 2000 to 2170.

The menu subroutine displays the function and provides the choices—to stop, to compute a table of function values, and to use the bisection method.

Lines 500 to 640 contain the subroutine that computes the table of values.

Lines 520 to 530 input a starting X value X0.

Lines 540 to 550 input an X step size DX.

Lines 580 to 610 compute and print the value of the function at 20 points, starting at X0 and incrementing by DX.

Lines 1000 to 1260 contain the bisection subroutine. This subroutine uses exactly the same commands as the bisection program you worked with in Exercise 31-1. Since you've already studied the bisection program, I left out most of the explanatory REM statements.

To start off the bisection method, you must find XL and XH such that the function is positive at one and negative at the other. To do this, use the table subroutine. Compute the function at various X points and look for the function values to change sign.

Exercise 31-18. Use BISECTION GENERAL to find the roots of the function (31-2), the one the program comes with, as follows:

1. Use the table option to compute some values of the function. Start at X = −10 and use a step size of 1.
2. Look at the function values you computed. If one of them is zero, you have found a root. Also look for places where the function changes sign. If you find points XL and XH such that F(XL) and F(XH) have different signs, use them as starting points for the bisection method.

   The function in Exercise 31-18 is a polynomial of degree three, so it can have at most three roots. Steps 1 and 2 above should have led you to three roots, so you found them all.

Exercise 31-19. What if you make a mistake and choose starting points XL and XH, which do not bracket a root? Try it on the function of Exercise 31-18 and see what happens.

Exercise 31-20. Use BISECTION GENERAL to find all roots of the polynomial

$$36X^3 - 1{,}197X^2 - 205X + 3{,}500$$

If the bisection method seems too slow for you, perhaps you'd prefer to use Newton's method to approximate the roots. Well, all you have to do is replace the bisection procedure with Newton's.

Exercise 31-21. Modify BISECTION GENERAL so that it uses Newton's method to approximate the roots. The bisection method resides in a subroutine that starts at line 1000. You just have to replace this subroutine with one that does Newton's method. Use the program NEWTON's as a guide. In fact, you can just copy over the same commands. You must also include a DEF statement near the beginning of the program for the derivative function. To test it out, try your program on a function whose roots you know, such as $X^2 - 2$.

Exercise 31-22. Use your general Newton's method program to find all roots of the polynomial

$$5X^3 - 18X^2 - 137X + 390$$

To get Newton's method started, you have to have a guess at a root. How do you come up with a guess? Use the table option to look at some function values. If you see the function change sign from one point to the next, you know there is a root in between, so you could choose some in-between point as a first guess.

Up to now, we've been rather relaxed about the first guess for Newton's method. That's because, for approximating square roots, the choice of first guess isn't too important. For some functions, however, it is. For example, consider the polynomial

$$2X^3 - 15X^2 + 24X + 17 \tag{31-3}$$

First, let's draw its graph in Exercise 31-23.

Exercise 31-23. Use your graphing program from Exercise 22-4 to draw the graph of function (31-3) as follows:

1. Draw the graph from $X = -1$ to $X = 5$.
2. Put in the X and Y axes.
3. Draw the vertical lines $X = 4$ and $X = 1$.

You should see from the graph that the function has a root somewhere between $X = -1$ and $X = 0$. Also, it has a local maximum at $X = 1$ and a local minimum at $X = 4$.

So where is the trouble with Newton's method? What if you choose $X = 1$ as a first guess? The tangent line at $X = 1$ is horizontal, so it will never hit the X axis, and you will not get any second point. The same thing will happen if you take $X = 4$ as a first guess. Also, if your first guess is to the right of the point $X = 1$, Newton's method may get confused and wander around near the point $X = 4$. You can see this geometrically if you imagine starting at $X = 3$, for instance, and see where the tangent lines take you. For more dramatic evidence, let's use Newton's method on this function and see what happens:

Exercise 31-24. Use your general Newton's method program on the function (31-3) with the following first guesses:

1. X = 1
2. X = 4
3. X = 3.5
4. X = 5
5. X = 0

You can see from this exercise that the choice of first guess can be crucial to the convergence of Newton's method. Would it help to use the table option to choose the starting point for Newton's method? Do Exercise 31-25 to see.

Exercise 31-25. Use the table option to compute some values of the function (31-3). Start at X = −10 and use step size 1.

From this table, you can see that there is a root between 0 and 1, but how could you have seen that X = 1 and X = 4 are dangerous points?

To summarize, with Newton's method your first choice must be close to a root. How close? Close enough so that there is no critical point between the root and your choice. (A *critical point* for a function F(X) is a point X0 where F′(X0) = 0 or F′(X0) doesn't exist.) Also, as you have seen in Exercise 31-24, your choice cannot be a critical point or the program will bomb out immediately.

Thus you see that there is a price for the speed of Newton's method. If you're not careful with your first choice, you may get in trouble. You can take a risk and race along with Newton's method or play it safe and plod along with the bisection method.

Exercise 31-26. Use either Newton's or bisection to find all the roots of the following polynomials:

1. $20X^3 - 243X^2 + 444X - 140$
2. $600X^3 - 198{,}518X^2 - 494{,}045X + 15{,}000$
3. $27X^3 + 603X^2 - 624X + 140$
4. $720X^3 - 242X^2 + 27X - 1$
5. $X^3 + 7X^2 + 11X + 5$

Now let's use our root search methods to study the arctangent function. The *arctangent* of a number N is a number X such that tan(X) = N. So the arctangent of N is a root of the function F(X) = tan(X) − N.

Exercise 31-27. Modify BISECTION GENERAL to approximate the arctangent of N for various N:

1. Insert a line before the function definition line to input the value of N.
2. Change the function to F(X) = TAN(X) − N.
3. Compute a table of function values for N = 2. Start at X = 0 and use 2 as step size. How many times does the function change sign?

You see from part 3 of this exercise that there appear to be many roots. To make the arctangent have unique values so that we have a well-defined function, we require that the root be chosen as close to 0 as possible.

Exercise 31-28. Approximate the arctangent of N = 2:

1. Use a table of values to find an interval of X values containing the root nearest X = 0.
2. Start the bisection process with this interval and approximate the root.
3. Compare with the computer's value, ATN(2).

Exercise 31-29. Approximate the arctangent of other numbers N and compare with the computer's ATN(N).

We end this chapter with two problems whose solutions require finding roots of nonpolynomial functions.

Exercise 31-30. (See Figure 31-3.) A central angle θ in a circle with center O

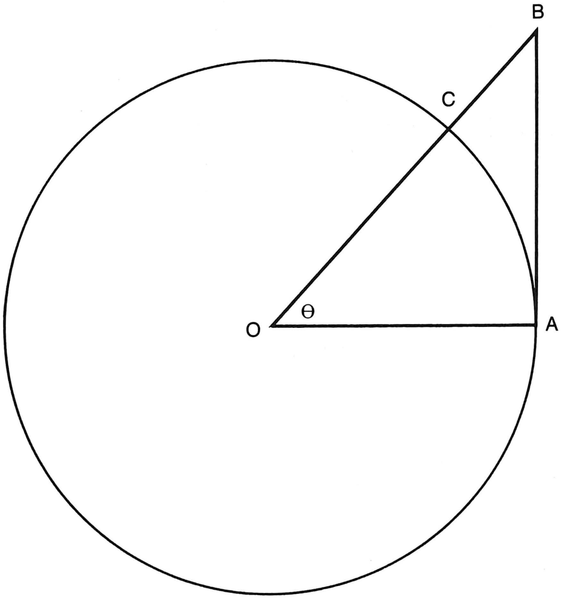


Figure 31-3 Circle, sector, and triangle

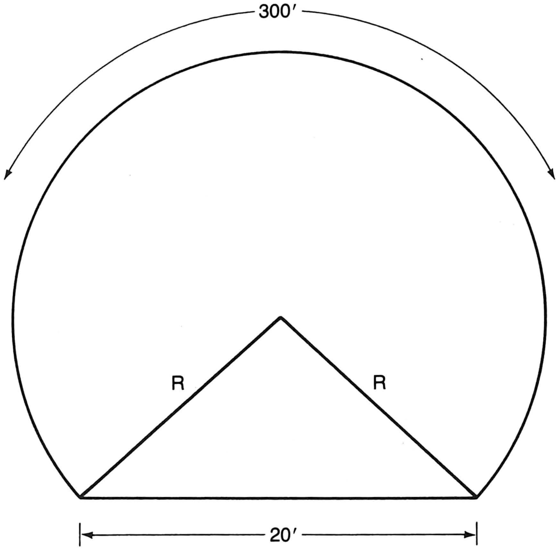


Figure 31-4 Chauncy's garden

determines a sector OAC and a right triangle OAB with one side AB on the tangent line to the circle at A. Find the angle θ so that the area of the triangle OAB is twice the area of the sector OAC.

Exercise 31-31. Chauncy Gardner wants to make a garden. He wants to use a straight 20-foot hedge as part of the boundary. The rest of the boundary he will make from a 300-foot piece of fencing. He will bend the fence into a circular arc as shown in Figure 31-4. To set up the fence, he must trace the circular arc on the ground.

1. What radius R must he use for the circular arc?
2. How can he find the center of the circle?

32

SERIES

In Chapter 31, we studied arithmetic approximation methods which the computer might be using in its SQR(X) function. Now we turn to some of the computer's other built-in functions, such as SIN(X) and EXP(X). The question is the same for these functions: How can the computer approximate the value of, say, the sine function using only arithmetic operations?

The answer may lie in the realm of *infinite series*. A series is a limit process that involves an infinite number of additions. We start with one of the simplest types of series—the geometric series.

First consider a geometric progression. In a geometric progression, to go from one number to the next, you multiply by a constant ratio R. So, if the first term is A, the terms in the progression must be

$$A,\ A*R,\ A*R^2,\ A*R^3,\ A*R^4,\ \ldots$$

We get a *geometric series* by adding up these terms: We start with the first term, A. When we add in the second term, A*R, we have

$$A + A*R$$

Adding in the third term, $A*R^2$ gives

$$A + A*R + A*R^2$$

After adding in the fourth term, $A*R^3$, the sum is

$$A + A{*}R + A{*}R^2 + A{*}R^3$$

We can, of course, continue adding in terms forever.

As we add up more and more terms of the series, we confront a convergence question: Do the sums we get approach a limit? If so, we say the series *converges*. If not, we say the series *diverges*.

We symbolize the entire series by writing down a few terms and adding on some dots to indicate that the addition continues indefinitely:

$$A + A{*}R + A{*}R^2 + A{*}R^3 + \ldots \tag{32-1}$$

To investigate the convergence question for the geometric series, let's have the computer add up the terms for us. We'll use a variable T for the individual terms and a variable S for the sum of the terms. Thus, T will take on the values A, A*R, $A{*}R^2$, $A{*}R^3$, and so forth. So, T will start off being equal to A. To get one term from the preceding one, we must multiply by R, so we will let T = T*R to get a new value for T in the program. The sum variable S will start with 0, and then we will add in the terms T. So, the procedure is as follows:

1. Initialize T = A and S = 0.
2. Add in T: LET S = S + T.
3. Get the next term T: LET T = T*R.

The program GEOMETRIC SERIES (see Listing 32-1) uses this procedure.

LISTING 32-1

```
100  REM  GEOMETRIC SERIES
110  HOME
120  PRINT "     THIS PROGRAM COMPUTES THE SUM OF"
130  PRINT : PRINT "THE TERMS OF THE GEOMETRIC SERIES"
140  PRINT : PRINT
150  PRINT "             2    3    4 "
160  PRINT "  A + AR + AR + AR + AR  + ...."
170  PRINT : PRINT
180  PRINT "    YOU SUPPLY THE FIRST TERM A"
190  PRINT : PRINT "AND THE RATIO R."
200  PRINT : PRINT
210  PRINT : PRINT "FIRST TERM A = ";: INPUT A
220  PRINT : PRINT "RATIO R = ";: INPUT R
230  HOME
240  PRINT : PRINT "              2    3    4 "
250  PRINT "  A + AR + AR + AR + AR  + ...."
260  PRINT "A = ";A
270  PRINT "R = ";R
280  PRINT : PRINT "HIT 'S' TO STOP, 'RETURN' TO CONTINUE."
290  PRINT : PRINT "# OF TERMS      SUM"
300  PRINT
310  VTAB 11
320  POKE 34,10: REM     PROTECT FROM SCROLLING
330  REM  INITIALIZE COUNTER FOR NUMBER OF TERMS:
```

```
340 K = 1
350  REM  INITIALIZE TERM VARIABLE:
360 T = A
370  REM  INITIALIZE SUM VARIABLE:
380 S = 0
390  REM  UPDATE SUM VARIABLE:
400 S = S + T
410  REM  PRINT NUMBER OF TERMS AND SUM:
420  PRINT K,S
430  REM  PAUSE:
440  GET ANS$
450  IF ANS$ = "S" THEN 510: REM    END
460  REM  COMPUTE NEW TERM:
470 T = T * R
480  REM  UPDATE COUNTER:
490 K = K + 1
500  GOTO 390: REM  DO IT AGAIN
510  POKE 34,0: REM  END SCROLLING PROTECTION
520  PRINT : PRINT "DO ANOTHER ONE";: INPUT ANS$
530  IF ANS$ = "N" THEN  END
540  HOME
550  GOTO 210
```

Lines 100 to 200 describe what the program does.

Lines 210 to 220 input the first term A and the ratio R.

Lines 230 to 320 print the series, the values of A and R, stopping instructions, and a header for the number of terms and the sums.

Lines 330 to 340 initialize the counter K for the number of terms in the sum.

Lines 350 to 360 initialize the term variable T to A.

Lines 370 to 380 initialize the sum variable S to 0.

Lines 390 to 400 add the term T into the sum S.

Lines 410 to 420 print the number K of terms and their sum S.

Lines 430 to 450 pause to see if you want to stop with this series. If so, control goes to the ending section, lines 510 to 550.

If you do not choose to stop in line 450, lines 460 to 470 compute a new term by multiplying the old T value by R. Then lines 480 to 490 update the counter and line 500 loops back for another round.

Exercise 32-1. Run GEOMETRIC SERIES to see how it works:

1. Use A = 1 and R = 1. The series is then

$$1 + 1*1 + 1*1^2 + 1*1^3 + \ldots$$

which is the same as

$$1 + 1 + 1 + 1 + \ldots$$

so you can check whether the program gives the correct sums.

2. Use A = 1 and R = −1. Do you get the sums you expect?
3. Use A = 1 and R = 0.5

In Exercise 32-1, neither of the series in parts 1 nor 2 converges. In part 1, the sums approach infinity, and in part 2 they oscillate between 1 and 0. The series in part 3 does converge. What is the limit of the sums in part 3?

Exercise 32-2. Run GEOMETRIC SERIES for A = 1 and various values for R, both positive and negative. Make a record of your R values and, for each choice, note whether the series seems to converge or not. From your observations, make a conjecture as to what R values will give a convergent series.

In Exercise 32-2, the first term is always A = 1. Does that affect the convergence of the series? Do the next exercise to see.

Exercise 32-3. Test the geometric series for convergence for various A and R. Make a record of your choices of A and R and whether the series for each seems to converge. What effect does the choice of A have on the convergence of the series?

You have seen in the exercises above that the geometric series converges for some choices of A and R and not for others. If it does converge for a certain A and R, can you predict what its limit will be?

Exercise 32-4. Find and write down the limiting value of the geometric series with A = 1 and each of the following R values. Although the R values are given as fractions 1/N, you must convert them to decimals for the program. Also, write down your limiting values, which the program will print as decimals, as (possibly improper) fractions. For ratio R = 1/N, find a formula for the limit in terms of N.

1. R = ½
2. R = ⅓
3. R = ¼
4. R = ⅕
5. R = ⅙

In Exercise 32-4, we kept A = 1 to make the limit formula easier to see. Now let's change A in Exercise 32-5.

Exercise 32-5. For A = 2, 3, and 4, write down the limit of the geometric series for each of the ratios of Exercise 32-4. Find a formula for the limit in terms of A and N.

Here is a summary of what you have seen in the exercises so far:

1. If |R| >= 1, the geometric series does not converge.
2. If |R| < 1, the series converges.
3. If R = 1/N, where N is a positive integer, the series converges to A*N/(N − 1).

The limiting value in part 3 of the summary above includes the quotient N/(N − 1), where R = 1/N. This quotient is usually written in terms of R as 1/(1 − R). To see that these quotients are equal, divide the numerator and denominator of the quotient N/(N − 1) by N. The result is

$$\frac{N}{N-1} = \frac{1}{1-\frac{1}{N}}$$

$$= \frac{1}{1-R}$$

because R = 1/N. Hence, part 3 of the summary can be rewritten as follows:

3′. For certain fractional values for R, the series converges to A/(1 − R).

In fact, for *every* R such that $|R| < 1$, the series converges to A /(1 − R). To check this out, modify GEOMETRIC SERIES in Exercise 32-6 so that it prints A/(1 − R).

Exercise 32-6.

1. Add a line to GEOMETRIC SERIES:

```
515 PRINT:PRINT"A/(1-R) = ";A/(1-R)
```

2. Run this version of the program for various A and R, with $|R| < 1$.
3. Does the series always converge to A/(1 − R)?
4. If R is positive, the sums increase toward the limit. How do the sums approach the limit if R is negative?

We have now seen that the geometric series converges to A/(1 − R), if $|R| < 1$. We summarize this fact in the following statement: If $|R| < 1$,

$$\frac{A}{(1-R)} = A + A*R + A*R^2 + A*R^3 + \ldots \qquad (32\text{-}2)$$

For our applications, it is important to think of the quotient A/(1 − R) as a function of R. Since we usually use X as a symbol for the variable in a function, let's change R to X in (32-2). Then it becomes: If $|X| < 1$,

$$\frac{A}{(1-X)} = A + A*X + A*X^2 + A*X^3 + \ldots \qquad (32\text{-}3)$$

Equation (32-3) is a rather remarkable statement. It equates a simple quotient, A/(1 − X), to an infinite sum!

The series on the right-hand side of equation (32-3) has terms of the form AX^n with the same coefficient A for each term. If we consider a series in which the coefficients are different for different terms, we write the series as

$$A_0 + A_1X + A_2X + A_3X^3 + A_4X^4 + A_5X^5 + \ldots \tag{32-4}$$

Such a series is called a *power series* in X. We also say that equation (32-3) gives a power series representation of the function A/(1 − X).

Why is it an advantage in computing to have a power series representation for a function? Well, a power series involves only addition and multiplication. Our function here, A/(1 − X), involves division, so by switching to a power series representation for it, we can avoid the division. This may not be very impressive, because the computer can do division easily anyway. But what if our function were, say, the sine function? This function is defined geometrically, either as the ratio of two sides of a triangle or as the y coordinate of a point on the unit circle. Neither definition gives a clue as to how to compute the function values. However, if we can find a power series representation for the function, we can approximate its values using only addition and multiplication.

Although a power series representation for a function reduces the computation to arithmetic, there are some complications. In the example above, for instance, the function, A/(1 − X), is defined for every X except X = 1. Unfortunately, the series in equation (32-3) converges only for $|X| < 1$, so this power series gives the function value at only some of the X values where the function is defined.

Now let's see if we can find a power series representation for F(X) = sin(X). To maintain a distinction between the geometrically defined sine function and the computer's approximation to it, we write sin(X) for the value of the sine function at an angle of X radians. We write SIN(X) for the value given by the computer's built-in sine approximating program.

We need to use some facts about the function:

$$F(0) = \sin(0) = 0 \tag{32-5}$$

Since $F'(X) = \cos(X)$,

$$F'(0) = \cos(0) = 1 \tag{32-6}$$

Since $F''(X) = -\sin(X)$, F(X) satisfies the differential equation

$$F''(X) = -F(X) \tag{32-7}$$

We assume we have a power series for F(X), namely,

$$F(X) = A_0 + A_1X + A_2X^2 + A_3X^3 + A_4X^4 + A_5X^5 + \ldots \tag{32-8}$$

and then we use equations (32-5) to (32-7) to see what the constants A_K must be.

For example, we know from (32-5) that F(0) = 0. On the other hand, if you plug X = 0 into (32-8) you get $F(0) = A_0$. Hence,

$$A_0 = 0$$

To make use of the information about the derivatives of F(X), we assume that we can differentiate a power series term by term. Then differentiating (32-8) shows that the first derivative of F(X) must be given by the series

$$F'(X) = A_1 + 2A_2X + 3A_3X^2 + 4A_4X^3 + 5A_5X^4 + \ldots \tag{32-9}$$

and the second derivative by

$$F''(X) = 2A_2 + 2*3A_3X + 3*4A_4X^2 + 4*5A_5X^3 + \ldots \tag{32-10}$$

If you plug $X = 0$ into the series in (32-9), you get $F'(0) = A_1$. From (32-6), $F'(0) = 1$, so we have

$$A_1 = 1 \tag{32-11}$$

Thus we know the first two coefficients in the power series for sin(X) must be 0 and 1.

Now we use the differential equation (32-7) to get the rest of the coefficients. We assume that two power series in X which represent the same function must have equal coefficients. Then (32-7) implies that each coefficient in the second-derivative series (32-10) is the opposite (additive inverse) of the corresponding coefficient in the original series (32-8):

$$2A_2 = -1*A_0 \tag{32-12}$$

$$2*3A_3 = -1*A_1 \tag{32-13}$$

$$3*4A_4 = -1*A_2 \tag{32-14}$$

$$4*5A_5 = -1*A_3 \tag{32-15}$$

and so forth. Since we know that $A_0 = 0$, equation (32-12) implies that $A_2 = 0$. Then (32-14) implies that $A_4 = 0$. In general, each coefficient with an even subscript is zero. Thus, the only terms that appear in the power series for sin(X) have odd degree.

Now let's see how to compute the coefficients of these odd terms. Equation (32-13) implies that

$$A_3 = \frac{-1*A_1}{2*3} \tag{32-16}$$

Continuing in this fashion, (32-15) implies that

$$A_5 = \frac{-1*A_3}{4*5} \tag{32-17}$$

In general,

$$A_{K+2} = \frac{(-1)*A_K}{(K + 1)*(K + 2)} \tag{32-18}$$

This shows how to get one coefficient, A_{K+2}, from the previous one, A_K: We must multiply by -1 and divide by $(K + 1)*(K + 2)$.

To compute one term $A_{K+2}X^{K+2}$ from the previous one, A_KX^K, we must not only get the new coefficient, but also the new power of X. But that is easy: We just have to multiply by X^2.

To put this in computer language, if T is the term of degree K, to compute the value of the next term, we must let

```
T = T*(-1)*X*X/((K+1)*(K+2))                    (32-19)
```

Finally, since we know the first term is X, we know how to initialize T and the degree variable K: We must let

$$T = X \quad \text{and} \quad K = 1 \tag{32-20}$$

Although we can compute the power series with just the information in equations (32-19) and (32-20), it is traditional to write out several terms of the power series. We already know that all the even coefficients are zero. Now let's find a formula for the odd coefficients. Since we know that $A_1 = 1$, (32-16) implies that

$$A_3 = \frac{-1}{2*3} = \frac{-1}{3!} \tag{32-21}$$

Thus we have the second odd coefficient. From (32-17) and (32-21), we get

$$\begin{aligned} A_5 &= -1*A_3*\left(\frac{1}{4*5}\right) \\ &= -1*\left(\frac{-1}{3!}\right)*\left(\frac{1}{4*5}\right) \\ &= \frac{1}{5!} \end{aligned} \tag{32-22}$$

In general, the odd coefficients, except for the alternating sign, are given by 1/(K!), where K is the degree of the term. Thus, the power series for sin(X) must be

$$X - \left(\frac{1}{3!}\right)X^3 + \left(\frac{1}{5!}\right)X^5 - \left(\frac{1}{7!}\right)X^7 + \ldots \tag{32-23}$$

Before we get back to computing, let's review what we've done. We assumed that a power series can be differentiated term by term and that two equal power series must have the same coefficients. (We also made at least one other assumption—see if you can spot it!) Then we used some properties of the sine function and its derivatives to show that if the sine function has a power series at all, it must be the one in (32-23). To put this discussion on a sound foundation, we would have to prove our assumptions about power series. Moreover, we would have to prove that the power series (32-23) really does converge to the sine function before we could use it safely in computations. Most calculus books that deal with power series do all of this proving.

Rather than get bogged down in the details of proofs, we will use the computer to investigate the power series (32-23) and compare its values to the computer's SIN(X). The program SINE SERIES (see Listing 32-2) uses equations (32-19) and (32-20) to compute the sums of the terms in the power series.

LISTING 32-2

```
100  REM   SINE SERIES
110  HOME
120  PRINT "     THIS PROGRAM COMPUTES THE SUMS OF"
130  PRINT : PRINT "THE TERMS OF THE POWER SERIES"
140  PRINT
150  PRINT : PRINT "     3       5       7"
160  PRINT "X - X /3! + X /5! - X /7!  + ..."
170  PRINT : PRINT
180  PRINT : PRINT "YOU SUPPLY THE VALUE FOR X:"
190  PRINT
200  INPUT "X = ";X
210  HOME
220  PRINT "HIT 'S' TO STOP, 'RETURN' TO CONTINUE."
230  PRINT : PRINT "X =   ";X
240  PRINT : PRINT "DEGREE              SUM"
250  PRINT
260  VTAB 7
270  POKE 34,5: REM   PROTECT FROM SCROLLING
280  REM  INITIALIZE DEGREE VARIABLE:
290 K = 1
300  REM  INITIALIZE TERM VARIABLE:
310 T = X
320  REM  INITIALIZE SUM VARIABLE:
330 S = 0
340  REM  UPDATE SUM VARIABLE:
350 S = S + T
360  REM  PRINT DEGREE AND SUM:
370  PRINT K,S
380  GET ANS$
390  IF ANS$ = "S" THEN 450: REM   END
400  REM  COMPUTE NEW TERM:
410 T = T * ( - 1) * X * X / ((K + 1) * (K + 2))
420  REM  UPDATE DEGREE:
430 K = K + 2
440  GOTO 350
450  POKE 34,0
460  PRINT : PRINT "COMPUTER'S SIN(";X;") = "; SIN (X)
470  PRINT : PRINT "DO ANOTHER ONE";: INPUT ANS$
480  IF ANS$ = "N" THEN  END
490  HOME
500  GOTO 200
```

Lines 100 to 200 print a description of the program and input the value for X.

Lines 210 to 270 print some instructions and a header.

Lines 280 to 290 initialize the degree variable K to 1 as required by equation (32-20).

Lines 300 to 310 initialize the term variable T to X as required by equation (32-20).

Lines 320 to 330 initialize the variable S, which will hold the sum of the terms, to S = 0.

Lines 340 to 350 update the sum S by adding in the term T.

Lines 360 to 370 print the degree K of the highest term and the sum S of the terms.

Lines 380 to 390 pause to see if you want to stop. If not, lines 400 to 410 compute the next term T according to (32-19).

Then lines 420 to 430 update the degree variable K and line 440 sends control back to line 350 for another round.

If you decide to stop in line 390, control is sent to line 450 which stops the scrolling protection.

Then line 460 prints the computer's value of SIN(X) and lines 470 to 480 give you the chance to evaluate the series for another X.

Exercise 32-7. Run SINE SERIES for various X values. Does the series seem to converge to SIN(X)? Can you find an X for which the series does not approach SIN(X)?

For many values of X, the sum values eventually become constant as the degree increases, and the final value for the sum is quite close to SIN(X). Now let's get a little more organized in our investigation of the series.

Exercise 32-8. Run SINE SERIES for each of the following X values. For each X, do enough steps that the sums stabilize to a final value. Write down the degree for which the sum first hits its final value. Also write down the value of SIN(X) and note to how many decimal places the final value agrees with SIN(X).

1. X = 0
2. X = −0.2
3. X = 0.5
4. X = 1
5. X = 2
6. X = 3
7. X = 5
8. X = 10
9. X = 15
10. X = 20
11. X = 25
12. X = 30
13. X = 50
14. X = 100

You see from the results of this exercise that the series converges rather quickly for X values close to 0. For X farther away from 0, it takes longer for the sums to converge. Moreover, for X close to 0, the sums converge to values very close to SIN(X), but for X farther away from 0, the final sum value is likely to be farther away from SIN(X). In fact, you may find the sums converging to limits outside the interval from −1 to +1. Those limits are ridiculous as approximations to the sine function, which you know takes on only values from −1 to +1.

For X close to 0, the series converges rapidly to the computer's value for SIN(X). Hence, the computer could well be using our power series in its SIN(X) program. But what does the computer do to avoid the large errors for X values that are not near 0? It could be using identities. (Yes, indeed, identities are useful!)

For example, the sine function has period 2π, so we have the identity

$$\sin(X) = \sin(X - 2\pi) \tag{32-24}$$

This means that we can subtract any multiple of 2π from X without changing the value of its sine. What good does this do us here? It allows us to shift X to the interval $-\pi < X \leq \pi$. For example, if $X = 4.7\pi$, we could let

$$X = X - 2*2\pi$$

Now $X = 0.7\pi$, and this X has the same sine as the original 4.7π. This new X is much closer to 0, so the power series should give a good value for its sine.

What if X is out of the interval in the other direction, say $X = -23.2\pi$? Then we can use the identity

$$\sin(X) = -\sin(-X) \tag{32-25}$$

By letting $X = -1*X$, we convert X to 23.2π. Now we shift it down by letting

$$X = X - 12*2\pi$$

Now $X = -0.8\pi$. After using the power series on -0.8π, we have to multiply the series value by -1 because of the identity (32-25).

The program SINE SHIFTER (see listing 32-3) is the same as SINE SERIES except that it has some added lines to do the shifting of X.

LISTING 32-3

```
100  REM    SINE SHIFTER
110  HOME
120  PRINT "     THIS PROGRAM COMPUTES THE SUMS OF"
130  PRINT : PRINT "THE TERMS OF THE POWER SERIES"
140  PRINT
150  PRINT : PRINT "     3       5       7"
160  PRINT "X - X /3! + X /5! - X /7!  + ..."
170  PRINT : PRINT
180  PRINT : PRINT "YOU SUPPLY THE VALUE FOR X:"
190  PRINT
200  INPUT "X = ";X
202  REM  RECORD INITIAL X:
204 XI = X
210  HOME
220  PRINT "HIT 'S' TO STOP, 'RETURN' TO CONTINUE."
230  PRINT : PRINT "X =   ";X
240  PRINT : PRINT "DEGREE              SUM"
250  PRINT
260  VTAB 7
270  POKE 34,5: REM   PROTECT FROM SCROLLING
275  GOSUB 1000: REM  REDUCE X
280  REM  INITIALIZE DEGREE VARIABLE:
290 K = 1
300  REM  INITIALIZE TERM VARIABLE:
310 T = X
```

```
 320  REM  INITIALIZE SUM VARIABLE:
 330 S = 0
 340  REM  UPDATE SUM VARIABLE:
 350 S = S + T
 360  REM  PRINT DEGREE AND SUM:
 370  PRINT K,S * SC
 380  GET ANS$
 390  IF ANS$ = "S" THEN 450: REM   END
 400  REM  COMPUTE NEW TERM:
 410 T = T * ( - 1) * X * X / ((K + 1) * (K + 2))
 420  REM  UPDATE DEGREE:
 430 K = K + 2
 440  GOTO 350
 450  POKE 34,0
 460  PRINT : PRINT "COMPUTER'S SIN(";XI;") = "; SIN (XI)
 470  PRINT : PRINT "DO ANOTHER ONE";: INPUT ANS$
 480  IF ANS$ = "N" THEN  END
 490  HOME
 500  GOTO 200
1000  REM   SUBROUTINE TO SHIFT X
1010  REM  EVALUATE COMPUTER'S PI:
1020 PI = 4 *  ATN (1)
1030  REM  INITIALIZE SIGN CHANGE VARIABLE:
1040 SC = 1
1050  REM   START SHIFTING X:
1060  REM    MAKE X >= 0:
1070  IF X < 0 THEN X =  - 1 * X:SC =  - 1 * SC
1080  REM     HERE, X >= 0
1090  REM   SEE IF X <= PI:
1100  IF X <  = PI THEN 1140: REM  SKIP SHIFT
1110  REM    HERE, X > PI, SO SHIFT:
1120 X = X - 2 * PI
1130  GOTO 1100: REM  SEE IF X <= PI
1140  REM   HERE, -PI < X <= PI
1150  RETURN
```

Because the shifting changes X, lines 202 to 204 record the initial value of X as XI. Then, at the end, line 460 prints the computer's value of SIN(XI).

Line 275 calls the subroutine, lines 1000 to 1150, which shifts X.

Lines 1010 to 1020 set PI to the computer's value for π.

Lines 1030 to 1040 initialize the sign-change variable SC to 1. We will change SC to -1 if we use identity (32-25). Later, line 370 multiplies the power series sum S by SC in order to correct S for any use of this identity.

If X is negative, line 1070 changes its sign and also changes SC.

Line 1100 tests whether X $\leq$ PI. If so, it is in the desired range, so no further shifting is necessary.

If X $>$ PI, line 1120 subtracts 2*PI from X to shift it to the left. Then line 1130 sends control back to line 1100, which tests this new X. This shifting and testing stop as soon as X $\leq$ PI.

When the shifting stops, we have X $\leq$ PI. Moreover, X $> -$PI. To see this, note that line 1120 shifts X only if X $>$ PI. For such X, X $-$ 2*PI $> -$PI, so the shifted X is always larger than $-$PI. Hence, the final value for X lies in the interval $-$PI $<$ X $\leq$ PI.

When the control returns from the subroutine to line 280, the program proceeds to compute the power series as usual for the shifted X value. Line 370 prints the power series sum S, modified as necessary by the sign-change variable SC.

Exercise 32-9. Use SINE SHIFTER to repeat Exercise 32-8. For what X values, if any, does this "shifty" version do better than the original one?

You see from Exercise 32-9 that shifting X to the interval from $-\pi$ to π before applying the power series gives values closer to the computer's SIN(X), particularly for X values far from 0. You can do even better, however. Once you have X satisfying $-\pi < X \le \pi$, you can use equation (32-25) to change the sign of any negative X, so you can shift to an X in the interval $0 \le X \le \pi$.

Once you have an X in this interval, you can shift to the interval $0 \le X \le \pi/2$ as follows: If $X \geqq \pi/2$, let $X = \pi - X$. Then $0 \le X \le \pi/2$. This final shift preserves the sine function because of the identity

$$\sin(X) = \sin(\pi - X) \tag{32-26}$$

Exercise 32-10. Extend the shifting subroutine of SINE SHIFTER so that its final X is in the interval from 0 to $\pi/2$.

Exercise 32-11. Use your modified version of SINE SHIFTER to repeat Exercise 32-8. For what values of X, if any, is this version better than SINE SHIFTER?

Before leaving the sine function, we should examine a part of the shifting operation you have probably been grumbling about. Lines 1110 to 1130 shift large X values to the left by repeatedly subtracting 2*PI from X. But you know, from Chapter 1, that such repeated subtracting will introduce round-off error. So the computer's shifted X may be a little off. Well, in Chapter 1, you saw a way to avoid this round-off error, so use it here:

Exercise 32-12.

1. Delete lines 1120 and 1130.
2. Insert new lines between 1110 and 1140 to do the shifting. First find the correct multiple of PI that must be subtracted from X and then subtract that multiple from X all at once.

Now we move on to the cosine function. We'll compute a power series for it and compare the power series value to the computer's value for COS(X). The development of a power series for the cosine function is entirely similar to what we did earlier for the sine function. If we let $F(X) = \cos(X)$, then

$$F(0) = \cos(0) = 1 \tag{32-27}$$

Differentiating the function gives $F'(X) = -\sin(X)$, so

$$F'(0) = -\sin(0) = 0 \tag{32-28}$$

Differentiating a second time gives $F''(X) = -\cos(X)$, so we have the differential equation

$$F''(X) = -F(X) \tag{32-29}$$

which is the same as equation (32-7), which we found for the sine function.

We assume that F(X) has a power series representation given by equation (32-8). Then equation (32-27) implies that the first coefficient is $A_0 = 1$. Also, equation (32-28) implies that the second coefficient is $A_1 = 0$.

Since the cosine function satisfies the same equation, equation (32-29), as the sine function did, equations (32-12) through (32-15) are also true for the coefficients of the cosine series. Thus, since $A_1 = 0$, equation (32-13) implies that $A_3 = 0$. Then, equation (32-15) implies that $A_5 = 0$. In general, each odd coefficient in the cosine series is zero.

For the even coefficients, we again have equation (32-18) to get one even coefficient from the previous one. Hence, again, equation (32-19) shows how to compute one term of the series from the previous one. Thus, you need to change only the initializations to convert the program SINE SERIES to one that computes the series for the cosine function.

Exercise 32-13. Modify SINE SERIES so that it computes the power series for the cosine function:

1. The first nonzero term in the cosine series is $A_0 = 1$. Hence, change the starting degree to $K = 0$, and the starting term to $T = 1$.
2. Change the ending so that it reports the computer's value for COS(X) rather than SIN(X).

Exercise 32-14.

1. Run your cosine series program for various X to see how well it does.
2. Check that this series also has trouble for X values far from 0.
3. Find identities for the cosine function that will allow you to shift X near to 0.
4. Modify your cosine series program so that it shifts X according to your identities from part 3.
5. Test out your "shifty" cosine program.

Since the cosine function satisfied the same differential equation, equation (32-7), as the sine function, we didn't need to change the way we computed the new term T in the sine series program. We're not so lucky with the other functions. However, we can still use information about their derivatives to see what to do.

Now we turn to the exponential function, $F(X) = e^X$. We have $F(0) = 1$, so in the series given by equation (32-8), $A_0 = 1$. Hence, the starting degree is

$$K = 0 \tag{32-30}$$

and the starting term is

$$T = 1 \tag{32-31}$$

Turning to the derivative, we find that $F'(X) = eX$, so our function satisfies the differential equation

$$F'(X) = F(X) \tag{32-32}$$

Hence, each coefficient in the first-derivative series (32-9) must be equal to the corresponding coefficient in the original series (32-8). Hence,

$$\begin{aligned} A_1 &= A_0 \\ 2*A_2 &= A_1 \\ 3*A_3 &= A_2 \\ 4*A_4 &= A_3 \end{aligned}$$

and so forth. From these equations, we get

$$\begin{aligned} A_1 &= A_0 \\ A_2 &= \frac{A_1}{2} \\ A_3 &= \frac{A_2}{3} \\ A_4 &= \frac{A_3}{4} \end{aligned}$$

and so forth. So, in general,

$$A_{K+1} = \frac{A_K}{K + 1} \tag{32-33}$$

This equation shows how to compute one coefficient from the previous one. To get the term $A_{K+1}X^{K+1}$ from A_KX^K, then, we must multiply by X to get the new power of X and divide by K + 1 to get the new coefficient. That is, if T is the term of degree K, to convert to the one of degree K + 1, we must let

```
T = T*X/(K + 1)                                    (32-34)
```

Since the degree of the term has increased by 1, we get the new degree by letting

$$K = K + 1 \tag{32-35}$$

Exercise 32-15. Modify the program SINE SERIES so that it computes the series for e^X. Compute the new term according to (32-34) and initialize K and T as in equations (32-30) and (32-31). Update the degree K by equation (32-35). Try your program on various X to see how close it comes to the computer's EXP(X). Could the computer be using this series in its EXP program?

Now let's look at the natural logarithm function, which we will write as log(X). Since log(0) isn't defined, it is easier to find a power series for log (X + 1) instead of one for log(X). Consider the function F(X) = log (X + 1). This function satisfies F(0) = log(1) = 0, thus $A_0 = 0$. The derivative of F is

$$F'(X) = \frac{1}{X + 1}$$

Thus we have the differential equation

$$1 = (X + 1)*F'(X) \tag{32-36}$$

If we substitute the series (32-9) for F′(X) in equation (32-36), we get

$$1 = (X + 1)(A_1 + 2A_2X + 3A_3X^2 + 4A_4X^3 + \ldots)$$

If we assume we can multiply out the right-hand side and collect terms, we get

$$1 = A_1 + X(A_1 + 2A_2) + X^2(2A_2 + 3A_3) + X^3(3A_3 + 4A_4) + \ldots \tag{32-37}$$

Hence, if we put X = 0 in the right-hand side, we get $A_1 = 1$. Since we already saw that $A_0 = 0$, we now know that the series starts with degree K = 1 and first term T = 1.

If we think of the left-hand side of equation (32-37) as a power series, all of its coefficients above the constant term are zero. For the equality in (32-37) to hold, then, all the higher coefficients on the right-hand side must also be zero. That is,

$$A_1 + 2A_2 = 0$$

$$2A_2 + 3A_3 = 0$$

$$3A_3 + 4A_4 = 0$$

and so forth. These equations give

$$A_2 = \frac{-1*A_1}{2}$$

$$A_3 = \frac{-1*A_2*2}{3}$$

$$A_4 = \frac{-1*A_3*3}{4}$$

In general, for $K \geqq 1$,

$$A_{K+1} = \frac{-1*A_K*K}{K + 1}$$

Hence, to get the term $A_{K+1}X^{K+1}$ from the previous one, A_KX^K, we must multiply by -1, X, and K, and divide by K + 1. Thus, if T is the term of degree K, to convert to the next one, we must let

```
T = -1*T*X*K/(K + 1)
```

Exercise 32-16. Modify SINE SERIES so that it computes the series for log(X + 1). Compare the series values to the computer's LOG(X + 1). Note how long it takes the series to converge. Do you think the computer's LOG function is using this series?

Your experience in Exercise 32-16 should convince you that the computer is probably not using that series in its LOG program. So what else could it be doing? You already have seen another possibility, the integral, in Exercise 30-20, part 7.

Exercise 32-17. Use the program INTEGRAL FUNCTION to approximate values of LOG(X + 1) as you did in Exercise 30-20. Is this integral method better than the series method?

As a final example, let's look at the arctangent function, which we will write F(X) = atn(X). This function satisfies

$$F(0) = \text{atn}(0) = 0$$

and

$$F'(X) = \frac{1}{1 + X^2}$$

Thus,

$$F'(0) = 1$$

and we have the differential equation

$$1 = F'(X) * (1 + X^2)$$

Exercise 32-18. Use the information above about the function atn(X) to find how to compute a power series for it. Modify SINE SERIES so that it computes your power series and compare the series values to the computer's ATN(X). Which is a better method of approximating the arctangent, this series or the integral? (For the integral method, see Exercise 30-20, part 6.)

33

HINTS AND ANSWERS

Chapter 3, Euclidean Algorithm

Ex. 3-12. Change line 130 to

```
130 FOR N2 = N1 TO BOUND
```

Chapter 4, Divisors

Ex. 4-1. Print N/K only if N/K <> K. Stop testing K when K*K > N.

Ex. 4-2. Use a variable SUM for the sum of divisors. When a divisor K or N/K is found, add it to SUM.

Ex. 4-3. The variable SUM of Ex. 4-2 gives the sum of all divisors, so SUM − N gives the sum of the proper divisors.

Ex. 4-6. The program PERFECT SEARCH, on the class disk, is a solution.

Chapter 5, Primes

Ex. 5-9. Put 2 in the table and then check only the odd numbers. When testing a number N for primeness, stop trying K as a divisor when K*K > N.

Chapter 6, Perfect Numbers Again

Ex. 6-1. Numbers 6, 28, and 496 all have the form M(2M − 1) where M is a power of 2.

Ex. 6-2. Start with $M = 2$. Let $N = M*(2*M - 1)$ and check whether N is perfect. Then let $M = M*2$ and check the new N.

Ex. 6-3. The prime factorizations reveal that a number of the form $N = M(2M - 1)$, where M is a power of 2, is perfect only when $2M - 1$ is prime.

Chapter 9, Means

Ex. 9-11. Each of the given sequences is an arithmetic progression. Also, the arithmetic mean of each sequence is ''nice.'' Namely, it is the arithmetic mean of any two terms symmetrically placed with respect to the middle of the sequence, for example, the first and last terms, the second and next-to-last terms, and so forth.

Ex. 9-12. Each of the sequences is a geometric progression and the geometric means are ''nice.''

Ex. 9-13. What is a harmonic progression? What is a harmonic? If a musical note has frequency F, its harmonics have frequencies 2F, 3F, 4F, ... If a string of length L produces the note with frequency F, the harmonics are produced by strings of lengths L/2, L/3, L/4, ... Run MEANS for various sequences of fractions for which the harmonic mean is ''nice.'' Formulate a definition of ''harmonic sequence'' and state and prove a theorem about the harmonic mean of such a sequence.

Chapter 10, Harmonic Mean of Fractions

Ex. 10-2. The harmonic mean of $1/m$ and $1/n$ is $2/(m + n)$.

Ex. 10-3. The harmonic mean of c/m and c/n is $2c/(m + n)$. That is, add numerators and denominators.

Ex. 10-4. To use the formula of Ex. 10-3, you must convert the fractions to ones with a common *numerator!*

Exs. 10-5 through 10-7. The harmonic mean is nice for sequences of the form

$$\frac{a}{b}, \frac{a}{b + d}, \frac{a}{b + 2d}, \frac{a}{b + 3d}, \ldots$$

That is, the reciprocals of the fractions form an arithmetic progression. For such a sequence, the harmonic mean is the middle term, if there is an odd number of terms, or the harmonic mean of symmetrically placed terms if there is an even number of terms.

Chapter 11, Square Roots

Ex. 11-11. We have fractions N/D, for

$$1 \leqslant D \leqslant DMAX$$

and

$$0 \leqslant N \leqslant D$$

The numerators $N = 0$ and $N = D$ always give the two fractions 0/1 and 1/1, respectively. For the rest of the fractions, we can assume

$$1 \leqslant N < D$$

Then we have all pairs N, D with

$$N < D \text{ and } 1 \leqslant D \leqslant DMAX$$

There are DMAX*(DMAX − 1) pairs with N different from D. Just half of these have N < D, so DMAX*(DMAX − 1)/2 is the number of pairs beyond the first two, 0/1 and 1/1. So, you can take SIZE to be 2 + DMAX*(DMAX − 1)/2.

Ex. 11-12. For a denominator D, to get N/D to equal LOW, you must have N = LOW*D. To get N/D = HIGH, you must have N = HIGH*D. So, for each D, N runs from LOW*D to HIGH*D. What can you use for the dimension of the arrays? The old SIZE worked for an interval of length 1, so for an interval of length (HIGH − LOW), multiply SIZE by (HIGH − LOW).

Chapter 12, Farey Sequences and Roots

Ex. 12-8. The maximum error is 1/(DL*DH).

Ex. 12-13. 1 in. = $\frac{1}{12}$ ft., so use 12 as maximum denominator. The area of the old garden was 37*69 = 2,553 square feet. So, the desired side length is $\sqrt{2{,}553}$. Since she can measure down to $\frac{1}{12}$ foot, approximate $\sqrt{2{,}553}$ by fractions with denominators no larger than 12. This gives

$$\frac{101}{2} < \sqrt{2{,}553} < \frac{556}{11}$$

A side length of $\frac{101}{2}$ feet gives a garden that is slightly too small, and a side length of $\frac{556}{11}$ gives one that is a bit too large.

Ex. 12-14. $\frac{1}{16}$ in. = $\frac{1}{192}$ ft., so use 192 as maximum denominator. Approximating $\sqrt{2{,}553}$ by fractions with denominators up to 192 gives

$$\frac{9{,}297}{184} < \sqrt{2{,}553} < \frac{2{,}779}{55}$$

Ex. 12-15. The area of Cleo's garden is $\pi * 11^2 = \pi * 121$ square asps. So Antony's garden area must be no greater than $\pi * \frac{121}{3}$. The radius of Antony's garden must be no greater than $\sqrt{121/3}$. Approximating this square root by fractions with denominators up to 20 gives

$$\frac{127}{20} < \sqrt{121/3} < \frac{108}{17}$$

Tell Antony he can use a radius of $^{127}\!/_{20}$ asps.

Ex. 12-16. By the Pythagorean theorem, the ladder must be at least

$$\sqrt{10^2 + 6^2} = \sqrt{136}$$

cubits long to reach the ledge. If the ladder extends until it touches the wall above the ledge, it rises $^{6}\!/_{10}$ cubit above the ledge (by similar triangles). Hence, the maximum length of the ladder is

$$\sqrt{11^2 + \left(6 + \frac{6}{10}\right)^2} = \sqrt{\frac{16{,}456}{100}}$$

So, find a fraction that is larger than $\sqrt{136}$ and smaller than $\sqrt{16{,}456/100}$.

The program MEDIANT ROOT SEARCH gives

$$\sqrt{136} < \frac{35}{3}$$

and

$$\frac{372}{29} < \sqrt{\frac{16{,}456}{100}}$$

Since

$$\frac{35}{3} < \frac{372}{29}$$

either $^{35}\!/_{3}$ or $^{372}\!/_{29}$ works.

Chapter 14, Periodic Decimals

Ex. 14-2. See the program LD AUTO BREAK, discussed after Ex. 14-6, for the decimal point printing.

Ex. 14-3. The fraction 1/N has a terminating decimal expansion if and only if the only prime factors of N are 2 and 5.

Ex. 14-4. The fraction 1/N is purely periodic if neither 2 nor 5 is a factor of N.

Ex. 14-5. The length of the nonrepeating part depends on how many times 2 and 5 divide into N. If 2 divides S times into N and 5 divides T times into N, then the length of the nonrepeating part is the larger of T and S.

Chapter 15, Brute-Force Pi

Ex. 15-1. This is a program where round-off error may cause you problems if you don't watch out. If you set S = 1/NUM, initialize XO = O, and compute the new X by XN = X0 + S, you are likely to get an error in computing YN = SQR(1 − XN*XN) for the final value of XN. This final value of XN should be 1, but round-off error may have thrown it off. To avoid this difficulty, compute XN by XN = K*S, for K = 0 to NUM.

Ex. 15-2. Note that the approximation is always smaller than π. This is to be expected because our approximation to the circumference is too small.

Ex. 15-3. For 100 points, it takes about 27 seconds on my Apple II+ to get the approximation 3.14129857. For 200 points, about 54 seconds to get 3.1414887.

Ex. 15-4. This may be done by starting with NUM = 200 (based on the approximation found in Ex. 15-2) and successively incrementing NUM by 1 until the approximation is larger than 3.1415.

Chapter 16, Archimedes' Pi

Ex. 16-3. For 24,576 sides, the low approximation is 3.14159264 and the high is 3.14159266. The computer's approximation to π is 4ATN(1) = 3.14159266, so the last 6 has been matched by the high approximation. In the further doublings, however, the high approximations come down to a 4 in the last place. So is the computer's value off in the last place? That's unlikely, so maybe there is something wrong with our method. In fact, the computer's low approximations *decrease* for the last couple of doublings, which shows something is wrong: The true values of the low approximations would be increasing!

Ex. 16-9. When the number of sides reaches 49,152, the approximations settle down to a 6 in the eighth place and stay there. So, this is a better method than the first one.

Chapter 17, Linear EquationsI

Ex. 17-10. The program EQUATIONS, on the class disk, is a solution.

Ex. 17-11 answers:

1. $X_1 = 3,\ X_2 = -1$
2. $X_1 = 2/7,\ X_2 = 3/5$
3. $X_1 = -43/14$
   $X_2 = -24/7$
   $X_3 = -11/14$

Chapter 18, Linear Equations II

Ex. 18-6. The system can be reduced to

$$X_1 = \frac{-1}{15}X_2 + \frac{1}{20}X_3 + \frac{1}{30}$$

then

$X_2 = 0$ and $X_3 = 1$ give $X_1 = 1/12$

$X_2 = 1$ and $X_3 = 0$ give $X_1 = -1/30$

Ex. 18-7. Here are two solutions:

$X_3 = 0$ gives $X_1 = -14/5$ and $X_2 = -43/5$

$X_3 = 1$ gives $X_1 = -19/5$ and $X_2 = -38/5$

Ex. 18-8. Answer: $X_1 = 1$, $X_2 = 1/2$, and $X_3 = 1/3$.

Ex. 18-9. This system is inconsistent.

Ex. 18-10. This system is indeterminate. Two of the solutions are:

For $X_3 = 0$, $X_1 = -5/7$ and $X_2 = 27/7$

For $X_3 = 1$, $X_1 = 19/7$ and $X_2 = 1/7$

Ex. 18-11. This system is inconsistent.

Ex. 18-12. This system has a unique solution: $X_1 = 3/4$, $X_2 = 7/2$, and $X_3 = 17/5$.

Ex. 18-13. This system is indeterminate. Two of the many solutions are:

For $X_3 = 0$, $X_1 = 67/34$ and $X_2 = -25/17$

For $X_3 = 1$, $X_1 = 44/34$ and $X_2 = -5/17$

Ex. 18-14. This system has a unique solution: $X_1 = 2/3$, $X_2 = -2$, $X_3 = 2/5$, and $X_4 = -1/4$.

Ex. 18-15. This system is inconsistent.

Ex. 18-16. This system is indeterminate. Two of the solutions are:

1. For $X_4 = 0$, $X_1 = 21/11$, $X_2 = 15/11$, and $X_3 = 19/11$.
2. For $X_4 = 1$, $X_1 = 14/11$, $X_2 = -1/11$, and $X_3 = -2/11$.

Ex. 18-17. This system has a unique solution: $X_1 = 1$, $X_2 = -1$, $X_3 = 1/3$, and $X_4 = 2$.

Ex. 18-19. Start with the system

$$\begin{array}{llllll}
(1) & X_1 & & & = & 1 \\
(2) & & X_2 & & = & 2 \\
(3) & & & X_3 & = & 3
\end{array}$$

Then add and multiply until you have three equations with no zero coefficients.

Chapter 20, Linear Equations and Round-Off Error

Ex. 20-1. Solution: $X = 17/105$ and $Y = 3/35$.

Ex. 20-2. Solution:

$X = 61/3$ and $Y = 58/3$.

Ex. 20-6. The solution in fractions is:

$$X = \frac{-839}{3} = -279.\bar{6}$$

$$Y = \frac{916}{3} = 305.\bar{3}$$

By method 1, I got $X = -260.96$ and $Y = 286.01$. By method 4, I got $X = -323.45$

and Y = 352.92. You may get something completely different, of course, if you use a sequence of operations different from mine.

Ex. 20-7. The solution in fractions is

$$X = \frac{893}{2} = 446.5$$

$$Y = \frac{-7{,}613}{16} = -475.8125$$

By method 1, I got X = 145.53 and Y = −154.03.

By method 3, I got X = −210.06 and Y = 227.61. How's that for different results? Does the system of Ex. 20-7 have a unique solution? Well, the coefficient matrix of the system is

$$\begin{pmatrix} 145 & 136 \\ 111 & 104 \end{pmatrix}$$

and the determinant of this matrix is

$$145*104 - 136*111 = -16$$

Since this determinant is not zero, a theorem of linear algebra says that the system does have a unique solution. Note that this determinant is small compared with the coefficients in the system of equations. This is a situation that often leads to bizarre round-off error results. Make up your own system with determinant near zero and see what happens!

Chapter 21, Lines

Ex. 21-9. B = 6.25 works well on my APPLE monitor.

Chapter 22, Graphs

Ex. 22-8. The local maximum is at X = −1.62857143 and Y = 9.08942274.

Ex. 22-9. The local minimum is at X = 0.614285714 and Y = −2.0900758.

Ex. 22-10. To find the point where the graph crosses the X axis, make a guess and draw the vertical line at your guess.

Ex. 22-11. Here are some important properties of these functions:

1. This function is zero at three distinct points. Does your graph show all three?
2. Y = sin(X) is periodic and oscillates between Y = −1 and Y = 1.
3. Y = tan(X) has vertical asymptotes.
4. Y = X*sin(X) oscillates between the lines Y = X and Y = −X.

5. $Y = e^X$ increases rapidly for positive X but approaches 0 for negative X. It is difficult to show both properties on one graph.

Ex. 22-12. The limit of sin(X)/X as X approaches 0 is 1, so defining Y to be 1 when X = 0 makes the function continuous. The usual proofs of the derivative formulas for sin(X) and cos(X) use this limit fact.

Chapter 24, Acceleration

Ex. 24-5. The paths should look like parabolas, which physics says is the correct path for a point moving according to our constant downward acceleration.

Ex. 24-11. To get acceleration in the X direction, change line 480 to give AX some nonzero value.

Ex. 24-12. To use a variable X acceleration AX, you need to compute AX before each computation of a new X velocity. For example, you could insert a line

```
xxx AX = Y
```

before the VX computation.

Chapter 25, Kepler and Newton

Ex. 25-5. The velocity that gives a circular orbit is smaller for points farther away from the sun. To develop a guess at a formula, try multiplying the X coordinates by the Y velocities. Is the product constant?

Ex. 25-6. This is a problem that *cannot* be solved on a computer! Suppose you start with X = 10, Y = 0, X velocity = 0, Y velocity = 10. At first, the Y coordinate increases. When (if!) the Y coordinate reaches a maximum value and starts to decrease, you know the planet is in orbit. To know that the planet is *not* in an orbit, you have to know that the Y coordinate will *never* start to decrease. So what are you going to do—let it run forever?

Ex. 25-9. To test whether Y has changed sign, see whether YN*YO <0.

Chapter 26, Parametric Equations

Ex. 26-10, part 1: X = T, Y = 2T − 2

Ex. 26-11. X = sin(T), $Y = \sin^2(T)$

Chapter 27, Polar Coordinates

Functions for graphs shown in Figure 27-5:

1. R = 3 − 5 sin(6T)
3. R = sin(2T) − cos(3T)

5. R = 3 − 2 tan(3T/5)
7. R = 2 + 4 sec(5T/4)
9. R = 10 − 2 sin(8T)
11. R = T

Chapter 28, Limits

Ex. 28-19.

1. The limit is *not* 0.

Ex. 28-20. The limit is, by definition, the number e.

Chapter 29, Derivatives

Ex. 29-8.

3. The derivative is 2|X|.

Chapter 30, Integrals

Ex. 30-14. The integral of X from 0 to B is $B^2/2$.
Ex. 30-15. The integral of X^2 from 0 to B is $B^3/3$.

Chapter 31, Root Searches

Ex. 31-20. One of the roots is between 33 and 34.
Ex. 31-26.

2. The roots are −2.5, 0.03, and 333.3.
4. The roots are ⅛, ⅑, ⅒.

Ex. 31-30. The area of the sector is

$$\frac{\theta * (OA)^2}{2}$$

The area of the triangle is

$$\frac{OA*AB}{2} = \frac{OA*OA*\tan(\theta)}{2}$$
$$= \frac{(OA)^2*\tan(\theta)}{2}$$

Equating twice the area of the sector to the area of the triangle gives

$$\theta*(OA)^2 = \frac{(OA)^2*\tan(\theta)}{2}$$

Canceling OA^2 leaves

$$\theta = \frac{\tan(\theta)}{2}$$

or

$$2\theta = \tan(\theta)$$

So, find a root of the function

$$F(X) = \tan(X) - 2X$$

Chapter 32, Series

Ex. 32-1, part 3. The limit is 2.

Ex. 32-4. For A = 1 and R = 1/N, the limit is N/(N − 1).

Ex. 32-6, part 4. For negative R, the sums oscillate—first larger than the limit, then smaller.

Ex. 32-18. To compute a power series for the arctangent, initialize K = 1 and T = X. Compute new T by

```
T = -1*T*X*X*K/(K + 2)
```

INDEX

A

Abundant number, 13
ACCELERATION, 157
Antiderivative, 228
Arctangent, 244
Arithmetic mean, 34, 38, 219

B

Binary form of numbers, 2
BISECTION, 231
BISECTION GENERAL, 240
Bisection method, 230
Bracketing fractions, 51
Bubble sort, 52

C

Cardioid, 185
Chain rule, 214
Conic section, 186
  directrix, 186
  eccentricity, 186
  focus, 186
Continuous function, 199
Convergent series, 248
Critical point, 244

D

Decimal:
  periodic, 82
  purley periodic, 81
  terminating, 80
Deficient number, 13
Derivative, 206
Derivative function, 209
DERIVATIVE FUNCTION, 210
DERIVATIVE LIMIT, 206
Directrix, 186
Divergent series, 248
Divisor, greatest common, 8
  proper, 13
DIVISORS, 12

E

Eccentricity, 186
Ellipse, 186
EQUATION GETTER, 108
Equation, pivot, 120
Equations:
  inconsistent system, 114
  indeterminate system, 114
  parametric, 171
EQUATIONS, 113
EQUATIONS CLEAR, 122
Eratosthenes, sieve, 15
Error, round off, 2, 127–33
Euclidean algorithm, 8

F

Factorization, prime, 19–22
Farey sequence, 58–60
Focus, 186
FRAC ARITH, 29
FRACTION BRACKETER, 68
FRACTION ROOT SEARCH, 56
FRACTION SORTER, 54
FRACTION SUBROUTINES, 43
Fractions:
  bracketing, 51
  reducing, 26
Function:
  continuous, 199
  critical point, 244
  derivative, 209
  inverse, 213
  root of, 230
Fundamental theorem of calculus, 228

G

GCD DEMO, 9
Geometric mean, 34, 38, 103
Geometric series, 247
GEOMETRIC SERIES, 248
GETTER, 30
GRAPHER, 142
Greatest common divisor, 8

H

Harmonic mean, 34, 38, 102
HCOLOR command, 134
HERON'S DEMO, 168
Heron's formula, 168
HGR command, 134
HPLOT command, 134
Hyperbola, 186

I

Inconsistent system of equations, 114
Indeterminate system of equations, 114
Infinite series, 247
INT function, 6
Integral, 215
INTEGRAL FUNCTION, 222, 263
INTEGRAL LIMIT, 216
Inverse function, 213

K

Kepler, Johannes, 162
Kepler's laws, 162
KEP1, 165

L

LD AUTO BREAK, 82
Left hand limit, 200
Limit, 197
  left hand, 200
  right hand, 200
  two sided, 200
LIMIT TESTER, 197, 213
LINES, 137
Lissajous curves, 177
Local maximum, 146
Local minimum, 146
LONG DIVISION, 78

M

Mean:
  arithmetic, 34, 38, 219
  geometric, 34, 38, 103
  harmonic, 34, 38, 102
MEANS, 38
Mediant, 60
MEDIANT ROOT SEARCH, 62
MULTIPLES, 3
MULTIPLY, 31
MULTIPLY AND PRINT, 32

N

Newton, Issac, 162
NEWTON'S, 237
Newton's method, 236
Number:
  abundant, 13
  binary form, 2
  deficient, 13
  perfect, 13, 24
  prime, 15
  prime factorization, 19–22
  proper divisor, 13
  quadruply perfect, 14
  triply perfect, 14
Numbers, twin prime, 22

O

ORDER, 52

P

Parabola, 186
Parameter, 171
Parametric equations, 171
PARAMETRIC EQUATIONS, 171
Perfect number, 13, 24
PERFECT SEARCH, 24
Periodic decimals, 82
Pivot equation, 120
Pivot variable, 120
Polar axis, 178
POLAR CONICS, 188
POLAR GRAPHER, 183
POLAR POINT PLOTTER, 179
Pole, 178
Power series, 252
Prime factorization, 19–22
Prime number, 15
Prime number theorem, 23
Primes, twin, 22
Proper divisor, 13
Purely periodic decimal, 81

Q

Quadruply perfect number, 14
Quotient, 3

R

Rectangle rule, 216
REDUCE, 27
Reducing fractions, 26
Remainder, 5
Right hand limit, 200
Root of a function, 230
Roses, 186
Rounding off formula, 131
Round off error, 2, 127–33

S

Scaling, graphics, 140
Series:
  convergent, 248
  divergent, 248
  geometric, 247
  infinite, 247
  power, 252
SIEVE, 18
SIEVE DEMO, 15
Sieve of Eratosthenes, 15
SINE SERIES, 255

Jon L. Sicks

INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS

This new book and accompanying disk were designed to be used in a mathematics course for prospective and in-service secondary teachers, in a high school computer-mathematics course, and as a resource for secondary mathematics teachers using microcomputers in their classes.

The microcomputer is a powerful tool for teaching mathematics, and **Jon L. Sicks** demonstrates new ways to consider traditional topics in order to make use of the computer. He shows how to approach such topics as: greatest common divisors, prime numbers, systems of linear equations, derivatives, and integrals. He also shows how to use computer graphics to graph functions and to model motion.

The author balances the text material between mathematics and programming. You need only a beginning knowledge of BASIC and an understanding of FOR-NEXT loops and IF-THEN statements.
Of special interest: A section of hints and answers for selected exercises. A section containing a discussion of each chapter. A description of the optional Solutions Disk.

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-503152-4

Sicks **INVESTIGATING SECONDARY MATHEMATICS WITH COMPUTERS** *Disk Included* Prentice-Hall

UNIVERSITY OF MASSACHUSETTS
AT AMHERST

Department of Mathematics and Statistics

Lederle Graduate Research Center
Amherst, MA 01003

9/11/87

Mrs. Waymire,

I enjoyed having lunch with you this summer and discussing mathematics and Teaching. Perhaps we can do it again on one of my next trips out.

Perhaps you can get access to a computer + try out some of the programs in my book.

Jon

P.S. Keep those tutoring charges up at a respectable level!!!